Version 10

11 janvier 2006

Préface

Ceci est la version 12 (automne 2005) du manuel des Séminaires Unix du Département d'Informatique et de Recherche Opérationnelle (DIRO) de l'Université de Montréal. Il diffère de la version 11 en incluant quelques corrections et mises à jour.

Le manuel a été entièrement réécrit à l'été 2002 pour le mettre à jour et migrer du format Microsoft Word au format LaTeX.

Afin d'alléger le texte, la forme masculine est utilisée et inclut la forme féminine partout où cela s'avère nécessaire.

La version la plus récente de ce document est toujours disponible à l'adresse suivante :

Manuel des Séminaires d'introduction à UNIX
`http ://www-etud.iro.umontreal.ca/~semunix/documentIRO`

URL

Pour organiser un séminaire Unix, contactez les responsables des séminaires Unix. Vous pouvez nous rejoindre par courrier électronique pour tous commentaires ou suggestions à :

`semunix@iro.umontreal.ca`

N'hésitez pas à nous faire part de toute correction, modification, clarification ou autre information que vous jugez pertinente ; vos commentaires nous sont précieux !

Répertoire de fichiers du manuel

Certains fichiers contenant divers exemples sont disponibles sur la page Web des séminaires UNIX. L'adresse de ce répertoire est

Fichiers d'exemple de ce manuel
`http ://www-etud.iro.umontreal.ca/~semunix/fichiers/`

URL

Historique

Hiver 1991 Des représentants de l'association des étudiants (AEIROUM [1]) et les administrateurs du réseau ont, d'un commun accord, décidé de mettre sur pied un projet de séminaires sur le système

[1] Association des Étudiants en Informatique et Recherche Opérationelle de l'Université de Montréal

d'exploitation UNIX. Ces séminaires permettent aux nouveaux étudiants (et aux anciens très souvent) de se familiariser rapidement avec le système qu'ils utiliseront pendant les quelques années à venir.

Avril 1991 Nous regroupions des volontaires pour développer les séminaires ; le but principal était, bien entendu, de produire des textes sur les différents sujets d'intérêts. Voici donc la liste des personnes qui ont contribué au tout premier document. Il s'agit, en ordre alphabétique, de messieurs (et madame) Myriam Beauchemin, Juan Berlie, Daniel Côté, Richard Dubois, Laurent Duperval, Pascal Forget, Eric Forthomme, Denis Garon, Benoît Jetté, Steven Lévesque, François McNeil, Jérôme Paré, Guy Pilon, Eric Sweeney, Jean Tessier, Daniel Guérard, Alexandre Guimond et Samy Touati de l'association des étudiants et de Jean-Claude Nadeau des services informatiques de l'Université. L'étape de correction, de mise en page et de fignolage fut effectuée par Myriam Beauchemin, Laurent Duperval, Benoît Jetté, Steven Lévesque, Guy Pilon, Jean Tessier et Jean-Claude Nadeau.

1991 - 2002 Le document a été révisé par d'autres acharnés et les conférenciers ont changé à quelques exceptions près.

Juillet 2002 Avec l'aide des étudiants des cycles supérieurs, le manuel fût complètement ré-écrit en format LaTeX.

2002-2005 Le document a été révisé par d'autres acharnés et les conférenciers ont changé à quelques exceptions près.

Auteurs et contributeurs

Ce manuel a été écrit par les étudiants suivants (quelques parties de ce manuel ont été importées de l'ancienne version) :

- Étienne Bergeron
- Éric Buist
- Luc Charest
- Jean-François Dufort
- Emric Epstein
- François Duranleau
- Eric Lesage
- Mathieu Miller
- Jonathan Sydney Roc
- Hervé Saint-Amand

Nous remercions également Bernard Derval de l'équipe du soutien technique pour ses commentaires et corrections.

Depuis 1994, les personnes suivantes ont contribué à la conception et l'évolution du manuel :

- Antoine Beaupré (étudiant)
- Paul Bratley (professeur)
- Danny Carrière (étudiant)
- Sébastien Constant (étudiant)
- Stéphane Doyon (étudiant)
- Marc-Antoine Drouin (étudiant)

- Massimo Fasciano (support technique)
- Stéphane Gaudreault (étudiant)
- Fabrizio Gotti (étudiant)
- Claude Goutier (support technique)
- Jonathan Laferrière (étudiant)
- Simon Labelle (étudiant)
- Guy Lapalme (professeur)
- Alexandre Le Bouthillier (étudiant)
- Steven Lévesque (étudiant)
- Jean-Louis Martineau (support technique)
- Patrick McNeil (étudiant)
- Jean-Claude Nadeau (étudiant)
- Sébastien Paquet (étudiant)
- Éric Plante (étudiant)
- Michel Robitaille (support technique)
- Raymond Thériault (étudiant)
- Yannick Thifault (étudiant)
- Huu Da Tran (étudiant)
- Marc Julien (étudiant)
- Gabriel Fillion (étudiant)

Copyright

Ce document est copyright ©1994-2005 (Édition 2005) par Gestion des Séminaires UNIX, tous droits réservés.

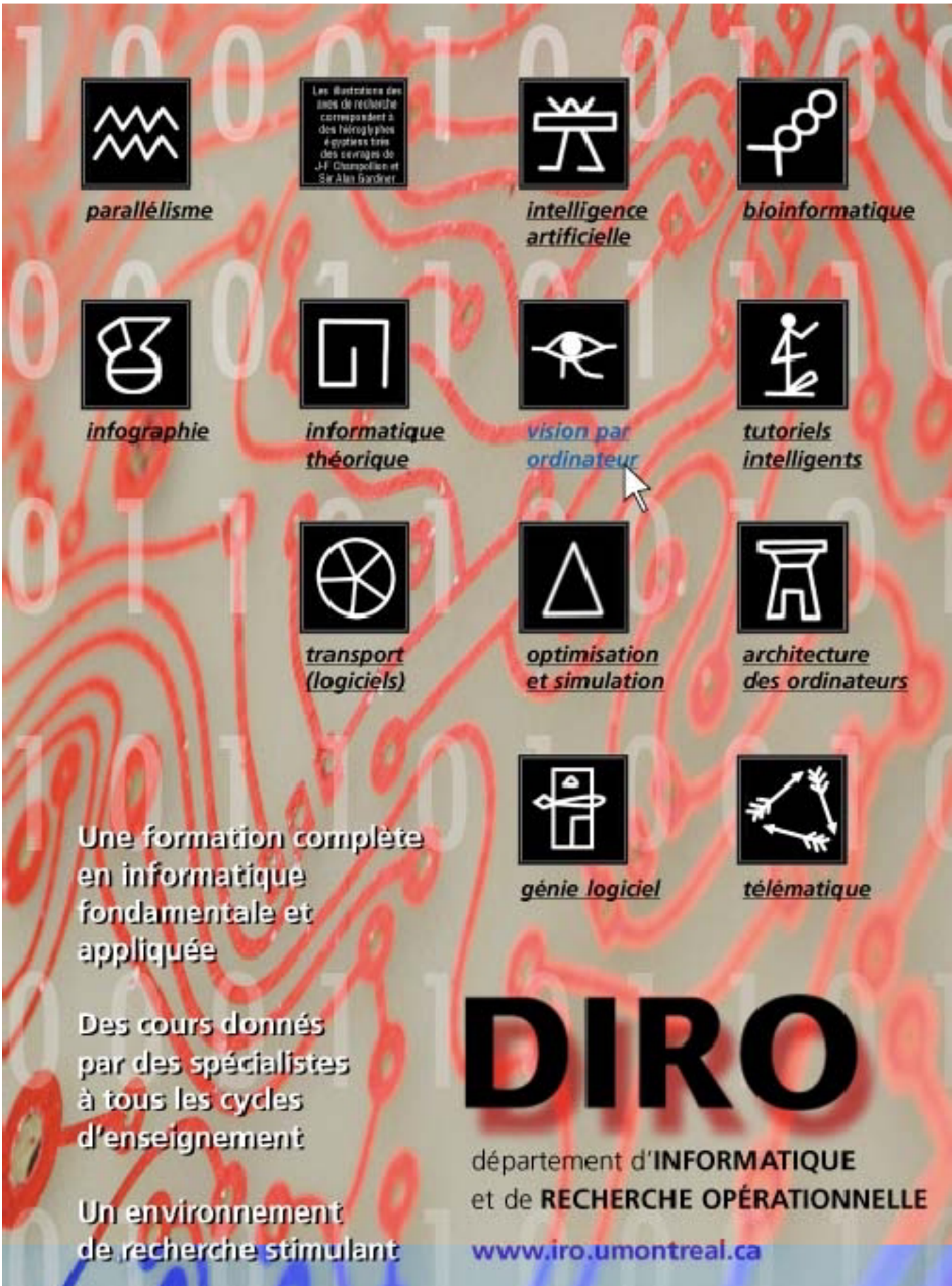
Contactez les responsables des séminaires si vous voulez redistribuer le manuel en tout ou en partie.

Commandites

Nous tenons sincèrement à remercier le support financier que le DIRO, le C.R.T., GIRO inc., l'APIIQ, le Camelot inc. et le service de polycopie de l'Université de Montréal nous ont apporté.

DIRO Département d'Informatique et de Recherche Opérationnelle

Service de polycopie de l'Université de Montréal


parallélisme
Les illustrations des axes de recherche correspondent à des hiéroglyphes égyptiens tirés des ouvrages de J-F Champollion et Sir Alan Gardiner
intelligence artificielle
bioinformatique
infographie
informatique théorique
vision par ordinateur
tutoriels intelligents
transport (logiciels)
optimisation et simulation
architecture des ordinateurs
génie logiciel
télématique
Une formation complète en informatique fondamentale et appliquée
Des cours donnés par des spécialistes à tous les cycles d'enseignement
Un environnement de recherche stimulant
DIRO
département d'INFORMATIQUE et de RECHERCHE OPÉRATIONNELLE
www.iro.umontreal.ca

Six succursales sur le campus • Des prix concurrentiels • Un service rapide

Notre défi ?
Vous impressionner !

Service de polycopie de l'U de M

- Reprographie
- Photocopie couleur
- Impression offset
- Assemblage et reliure
- Traitement postal
- Impression numérique
- Reproduction de photos (sur chandails, cotons ouatés, tapis à souris, casse-tête, cartes de souhaits et tasses).

Tél. : (514) 343.6410
Téléc. : (514) 343.2132
www.polycop.umontreal.ca

Pavillon principal	**N-315**
André-Aisenstadt	4309
3200 Jean-Brillant	B-2375
Marie-Victorin	C-259
Marguerite-d'Youville	1091
Médecine vétérinaire	2122

Université de Montréal

Table des matières

1 Le DIRO en quelques étapes rapides **15**
- 1.1 Trouver son nom d'usager et son mot de passe . . . 15
- 1.2 Les laboratoires publics . . . 15
- 1.3 Code d'éthique des laboratoires . . . 16
- 1.4 Comment se brancher . . . 17
- 1.5 Changer son mot de passe . . . 18
- 1.6 Au travail ! . . . 18
- 1.7 Se débrancher . . . 19

2 Les bases de l'informatique **21**
- 2.1 Les composantes d'un ordinateur . . . 21
  - 2.1.1 Le processeur . . . 21
  - 2.1.2 La mémoire . . . 22
  - 2.1.3 Le disque dur . . . 22
  - 2.1.4 Clavier et souris . . . 23
- 2.2 Le traitement de données . . . 24
  - 2.2.1 Qu'est-ce qu'un programme ? . . . 24
  - 2.2.2 Qu'est-ce qu'un fichier ? . . . 25
  - 2.2.3 Les répertoires . . . 26
  - 2.2.4 Qu'est-ce qu'un logiciel ? . . . 26
  - 2.2.5 Qu'est-ce qu'un système d'exploitation ? . . . 26
- 2.3 La communication entre ordinateurs . . . 27
  - 2.3.1 Les réseaux . . . 27
  - 2.3.2 L'Internet . . . 27
  - 2.3.3 Le World Wide Web . . . 28
  - 2.3.4 Le courrier électronique . . . 28

3 Les bases de UNIX **31**
- 3.1 Philosophie . . . 31
- 3.2 Concepts de base . . . 32
  - 3.2.1 Majuscules vs. minuscules . . . 32
  - 3.2.2 Le terminal . . . 32
- 3.3 Les usagers . . . 32
  - 3.3.1 Attributs d'un compte . . . 33

3.3.2 Les groupes d'usagers . . . 34
3.3.3 Le superusager . . . 34
3.4 Le système de fichiers . . . 34
3.4.1 Les types de fichiers sous UNIX . . . 34
3.4.2 Accès à un fichier . . . 35
3.4.3 Les droits d'accès . . . 37
3.5 Les commandes de base . . . 39
3.5.1 La syntaxe générale d'une commande . . . 39
3.5.2 pwd . . . 40
3.5.3 cd . . . 40
3.5.4 ls . . . 41
3.5.5 mkdir . . . 43
3.5.6 rmdir . . . 43
3.5.7 cp . . . 43
3.5.8 mv . . . 44
3.5.9 rm . . . 44
3.5.10 chmod . . . 45
3.5.11 echo . . . 47
3.5.12 man . . . 47
3.5.13 info . . . 48
3.5.14 Autres commandes . . . 48
3.6 L'environnement KDE . . . 49
3.6.1 Initialisation . . . 49
3.6.2 Aperçu du *desktop* . . . 49
3.6.3 Les *desktops* virtuels . . . 50
3.6.4 Opérations copier-coller . . . 50
3.6.5 Raccourcis clavier . . . 50
3.7 Obtenir de l'aide . . . 51
3.7.1 Que faire si ca plante ? . . . 51

4 Logiciels disponibles **53**
4.1 Internet et WWW . . . 53
4.1.1 Navigateurs Web . . . 53
4.1.2 Courrier électronique et News . . . 57
4.1.3 Transfert de fichiers . . . 59
4.1.4 Connexion à distance . . . 62
4.1.5 Autres outils . . . 63
4.2 Éditeurs de texte . . . 64
4.2.1 Emacs . . . 64
4.2.2 vi . . . 70
4.2.3 Pico . . . 80
4.2.4 Pilot : Gestionnaire de fichier . . . 82
4.2.5 Autres . . . 82
4.3 Bureautique . . . 82
4.3.1 OpenOffice . . . 82
4.3.2 LaTeX . . . 83

4.4 Archivage et compression . . . 83
4.4.1 gzip . . . 83
4.4.2 tar . . . 83
4.4.3 zip . . . 84
4.5 Imprimer . . . 85
4.5.1 Imprimer une page web dans Firefox . . . 85
4.5.2 Imprimer un fichier PDF avec Acrobat Reader . . . 85
4.5.3 Autres utilitaires . . . 86
4.5.4 En cas de problème . . . 87
4.6 Commandes utiles . . . 89
4.6.1 pager : less et more . . . 89
4.6.2 du . . . 89
4.6.3 dos2unix, unix2dos et mac2unix . . . 89
4.6.4 gimp . . . 90
4.7 Le compte casino . . . 90

5 Le réseau du DIRO **93**
5.1 Politiques d'utilisation . . . 93
5.1.1 Les droits des utilisateurs . . . 93
5.1.2 Les devoirs des utilisateurs . . . 94
5.2 Imprimer au DIRO . . . 96
5.2.1 L'imprimante payante . . . 96
5.2.2 Combien ça coûte, comment payer . . . 96
5.3 Les machines . . . 97
5.3.1 Stations de travail . . . 97
5.3.2 Serveurs . . . 97
5.4 Les commandes spécifiques au DIRO . . . 97
5.4.1 remise . . . 97
5.4.2 notes . . . 98
5.4.3 inclure . . . 98
5.5 Les adresses électroniques et Web des cours . . . 99

6 Pour en savoir plus **101**
6.1 Historique de UNIX . . . 101
6.1.1 UNIX . . . 101
6.1.2 Linux . . . 103
6.2 Concepts avancés de UNIX . . . 104
6.2.1 Les interpréteurs de commandes . . . 104
6.2.2 Les variables d'environnement . . . 114
6.2.3 Flots d'entrée/sortie . . . 114
6.2.4 Manipulation de fichiers . . . 116
6.2.5 Contrôle de processus . . . 119
6.2.6 X Window . . . 120
6.2.7 Expressions Régulières (ER) . . . 121
6.3 Comment travailler de chez soi . . . 121
6.3.1 Préalables . . . 121

6.3.2 Se connecter via SSH . . . 122
6.3.3 Accès à l'interface graphique . . . 124
6.3.4 Comment transférer des fichiers . . . 126
6.3.5 Comment consulter son courrier électronique . . . 127
6.3.6 Cygwin, une couche UNIX au-dessus de Windows . . . 128
6.3.7 Magellan . . . 130
6.4 Comment installer Linux chez soi . . . 135
6.4.1 Quelle distribution choisir ? . . . 135
6.4.2 Se procurer Linux . . . 138
6.4.3 Préparatifs . . . 140
6.4.4 Réaménagement du disque dur . . . 142
6.4.5 Installation . . . 143
6.4.6 Configuration . . . 144
6.5 Internationalisation . . . 145
6.5.1 Langue par défaut . . . 145
6.5.2 Configuration du clavier . . . 145
6.5.3 Alphabets non romains . . . 146

Chapitre 1

Le DIRO en quelques étapes rapides

Bienvenue au DIRO !

Ce premier chapitre constitue un tour d'horizon rapide pour les nouveaux usagers ; l'information qu'il contient permet de se mettre rapidement au travail. Il ne s'agit pas ici d'expliquer le fonctionnement détaillé de l'environnement DIRO, comme il sera fait dans les chapitres suivants, mais plutôt de permettre aux étudiants de se mettre au travail le plus vite possible.

Il est fortement recommandé de lire aussi le *« Bref guide pour l'étudiant du DIRO »* à cette adresse Web :

Bref guide pour l'étudiant du DIRO
`http ://support.iro.umontreal.ca/nouvetud.shtml`

URL

1.1 Trouver son nom d'usager et son mot de passe

Pour travailler sur un des postes public du département, vous aurez besoin de votre nom d'usager (*login*) et de votre mot de passe initial. Vous les trouverez au deuxième étage du pavillon André-Aisenstadt (voir figure 1.1), affichés sur un babillard dans le couloir près du secrétariat.

Ayant pris en note votre nom d'usager et votre mot de passe initial, vous pouvez vous diriger vers un des laboratoires, où se trouvent les postes de travail publics.

1.2 Les laboratoires publics

Le département d'informatique met à la disposition de ses étudiants quatre laboratoires qui sont ouverts **24 heures sur 24, 7 jours sur 7**. Ces laboratoires se trouvent au pavillon de mathématiques et d'informatique, le pavillon André-Aisenstadt (voir figure 1.1), dans les locaux **1340** (premier étage), **3185**, **3189**, et **3311** (troisième étage). Il est à noter que le local 3181, malgré sa proximité, est réservé aux étudiants des cycles supérieurs, et non à ceux du baccalauréat.

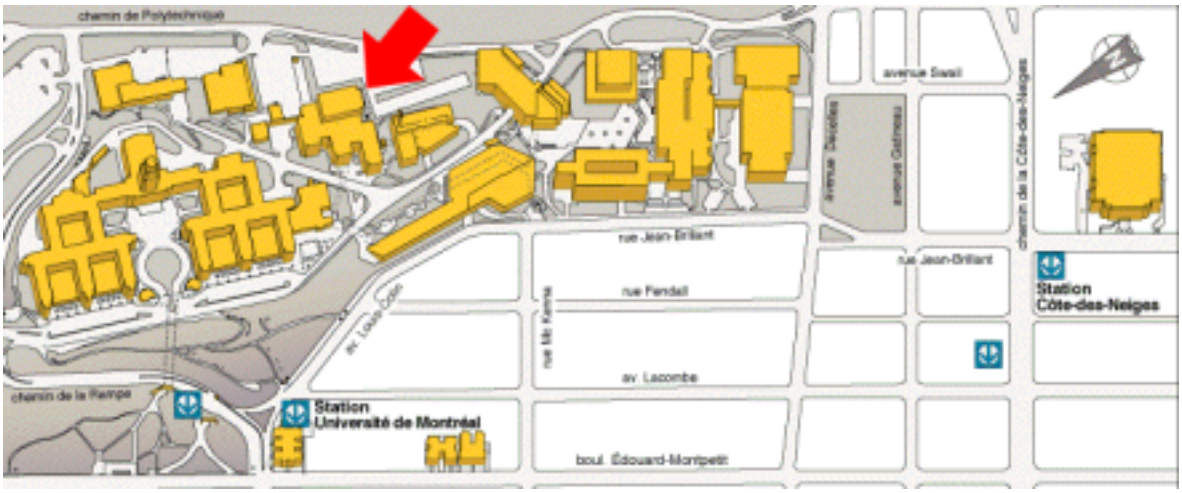


FIG. 1.1 – Pavillon André-Aisenstadt, 2920 chemin de la Tour

Notez que le local 1340 sert souvent à donner des cours ou des séances de travaux pratiques. Quand la salle est occupée par un cours, seuls les étudiants du cours sont admis. Une feuille indiquant que la salle est occupée est alors généralement placée sur la porte du local. Il en est de même pour les locaux 3185 et 3189, bien que cela soit plus rare.

1.3 Code d'éthique des laboratoires

Vous vous devez de respecter les règles d'utilisation des environnements informatiques de l'Université, disponibles à cette adresse :

URL

Règles d'utilisation des environnements informatiques de l'Université de Montréal
`http ://www.secgen.umontreal.ca/pdf/reglem/francais/sec_40/ges40_22.pdf`

La direction du DIRO insiste sur les points suivants :

- **Interdiction**
  - d'utiliser le compte usager d'une autre personne ;
  - de divulguer son mot de passe ;
  - formelle d'installer des logiciels ;
  - interdiction d'utiliser **abusivement le réseau**, soit en employant des logiciels de partage de fichiers (Kazaa, Morpheus, eDonkey, etc.), en laissant tourner un serveur (http, etc.) ou en transférant de grands fichiers (audio, vidéo, etc.) ;
  - de manger ou de boire dans les salles de stations de travail ;
  - de travailler dans un autre laboratoire que le sien sans la permission expresse d'un professeur ;
- **Respect**

  - de la propriété des informations et du droit d'auteur : ne pas copier ou rendre accessible les logiciels, les fichiers d'œuvres littéraires, scientifiques, musicales ou visuelles ;
  - de l'intégrité matérielle et logicielle des systèmes ;
- **Utilisation exclusive** des infrastructures à des fins d'enseignement et de recherche ;
- Un étudiant n'occupe qu'une seule station à la fois et se débranche s'il s'absente pour plus de quelques minutes ;
- L'étudiant doit présenter sa **carte d'étudiant** lorsqu'il lui est demandé de s'identifier.

Pour plus de détails, voir la section 5.1.

1.4 Comment se brancher

Comme il est expliqué plus loin dans ce manuel, vous devez entrer un nom et un mot de passe à chaque fois que vous voulez travailler sur un système UNIX(tous les détails dans la section 3.3). Les postes de travail du DIRO ne font pas exception à cette règle.

Attention, les nouveaux comptes créés au département doivent être activés avant de pouvoir vous brancher aux ordinateurs des laboratoires. Ceci sert à prévenir l'intrusion en utilisant les mots de passes par défaut. Vous devez donc activer votre compte (signaler au système votre présence) grâce à votre UNIP. Vous devez vous *logger* sur l'ordinateur de changement de mots de passe au local 3189 ou sur le site du support technique. Pour plus de détails sur le mot de passe et l'activation ainsi que pour trouver le lien qui permet l'activation, vous pouvez visiter le lien :

Détails sur le mot de passe et l'activation des comptes au DIRO
`http ://support.iro.umontreal.ca/motdepasse/`

URL

Avant de vous présenter à un poste de travail, vous devez imaginer le mot de passe que vous utiliserez. Comme la majorité des intrusions dans le système résultent d'un choix de mot de passe trop facile à deviner, il est **primordial** que vous choisissiez un mot de passe original. Il est important de répéter, parce que cette règle est trop souvent ignorée, que **vous ne devez pas choisir un mot de passe qui dépend d'informations personnelles**. Ne prenez pas un mot de passe contenant le nom de votre amoureux(se), ni celui de votre chien ou chat, ni votre adresse, etc. Ne prenez pas non plus un mot déjà existant. Votre mot de passe devrait idéalement être une combinaison de lettres, de chiffres et de symboles complètement inusitée.

Maintenant que vous connaissez votre nom d'usager et que vous avez imaginé votre mot de passe, installez-vous à l'un des postes de travail publics.

Si vous êtes dans le local 1340, une fenêtre montrant les choix de réseau vous est présentée. Choisissez **Cette_Station**.

Vous êtes maintenant face à une boîte d'identification similaire à celle présentée dans la figure 1.2. Entrez votre nom d'usager dans le champ *login* et votre mot de passe dans le champ *password*.

Assurez-vous que la touche *« fixmaj »* (*Caps Lock*) n'est pas enfoncée quand vous entrez vos nom et mot de passe. Le système différencie les majuscules des minuscules.

Attention

Si vous avez entré l'information correctement la boite d'identification disparaît. Vous êtes maintenant branché au système, ou, en bon jargon, *« loggué »*.

La première fois que vous vous branchez, le système vous demandera d'ajuster certains paramètres. Assurez-vous de lire la section 3.6.1 pour plus d'explications et pour des indications

FIG. 1.2 – Boite d'identification

concernant les options à choisir.

1.5 Changer son mot de passe

La première fois que vous vous branchez vous **devez** changer le mot de passe qui vous a été attribué car une tierce personne pourrait le trouver et l'utiliser.

Le programme vous demande d'entrer votre nouveau mot de passe deux fois. Il se peut que le programme refuse votre nouveau mot de passe s'il le juge trop facile à deviner. Réessayez jusqu'à ce que le programme accepte le mot de passe et affiche *« Le mot de passe UNIX a ete change »*.

Un mot de passe c'est comme une brosse à dents : on l'utilise fréquemment, on le change régulièrement et surtout on ne le prête à personne. N'oubliez pas de le changer régulièrement (une fois par session) avec la commande `chmp`.

Vous êtes maintenant prêt à découvrir et utiliser votre compte UNIX.

1.6 Au travail !

Afin de profiter pleinement de votre environnement de travail, nous vous recommandons de lire le présent manuel au complet, ou du moins les chapitres 3, 4 et 5.

De plus, vous devez prendre connaissance des politiques d'utilisation de votre compte (section 5.1) avant de l'utiliser.

Voici tout de même quelques pointeurs vers les sections les plus utiles :

- Pour naviguer sur le Web, voyez la section 4.1.1 ;
- Pour envoyer et lire des messages électroniques, voyez la section 4.1.2 ;
- Pour travailler avec des fichiers Office, voyez la section 4.3 ;
- Pour des détails sur l'environnement de travail KDE, voyez la section 3.6 ;
- Pour de l'aide générale, voyez la section 3.7.

Vous disposez de 60Mo d'espace disque pour vos fichiers personnels. Voyez la section 4.6.2 pour voir comment gérer votre espace disque.

Si vous avez perdu ou oublié votre mot de passe, voyez la page du support technique :

URL

Mots de passe perdus : réinitialisation
`http ://support.iro.umontreal.ca/motdepasse/`

1.7 Se débrancher

Quand vous avez terminé d'utiliser l'ordinateur, il est très important de ne pas oublier de se débrancher. Pour vous débrancher (*« délogguer »* en bon jargon), cliquez sur l'icône rouge portant un symbole d'interrupteur sur la barre au bas de l'écran (voir figure 1.3).

FIG. 1.3 – Se débrancher

Attention!

NE COUPEZ JAMAIS L'ALIMENTATION ÉLECTRIQUE DE L'ORDINATEUR !

Les systèmes de fichiers sur une machine UNIX contiennent d'habitude des données qui ne sont pas sauvegardées sur disque ; des dommages au système peuvent résulter d'une coupure brusque de l'alimentation.

Information

Si vous éprouvez des problèmes à vous débrancher, voyez les conseils de la section 3.7.1
.

Chapitre 2

Les bases de l'informatique

Ce chapitre vise à donner aux étudiants qui ont eu peu ou pas de contacts avec l'informatique les bases nécessaires à tout travail employant un ordinateur. Les concepts vus ici sont très généraux et ne s'appliquent pas uniquement au DIRO, ni à UNIX. Ce sont des notions globales que doit posséder tout informaticien.

Les étudiants qui sont déjà habitués au travail informatique peuvent survoler cette section rapidement, car elle ne sera pour eux que de la révision. Cette section devrait aider les étudiants qui ne sont pas familiers avec les concepts de base à comprendre le reste de ce document.

2.1 Les composantes d'un ordinateur

Voyons d'abord quelles sont les différentes pièces qui forment un ordinateur.

Comme vous le savez peut-être déjà, nous pouvons diviser la logique d'une machine en deux parties : le **matériel** et le **logiciel**. Voyons d'abord le matériel.

Cette section n'est qu'un survol des différentes pièces : le cours **IFT1214** donne une description plus formelle et plus détaillée des diverses composantes d'un ordinateur. Ce chapitre n'explique pas le fonctionnement interne des pièces, mais décrit plutôt leurs rôles et leurs relations.

2.1.1 Le processeur

Le **processeur** est la pièce la plus importante dans un ordinateur. C'est la puce électronique qui est le cerveau de votre machine. Lorsqu'un ordinateur exécute un programme, le processeur est responsable de lire chaque instruction une après l'autre et d'effectuer ce qui lui est demandé.

Une puce est une petite boite noire avec des pattes qu'on peut souder. Ne pas confondre avec l'insecte.

Information

Lorsqu'on parle du processeur, on parle souvent de **Hz** (Hertz) qui est une unité de mesure sur la vitesse d'exécution du processeur. En général, plus votre *CPU* (*central processing unit*, processeur) a de Hz, plus il est rapide. Les processeurs vendus sur le marché en 2002 atteignent parfois 2 GHz (2

milliards de Hz). Les processeurs sont fabriqués par des compagnies connues telles que Intel, AMD, Sun, Motorola, etc.

Attention

Le Hz est une mesure parmi tant d'autres et ne constitue pas en soi une certitude de vitesse. Plus de détails dans les cours d'architecture.

Si nous faisions une analogie entre un ordinateur et l'étudiant moyen, la chambre de l'étudiant serait le boîtier de l'ordinateur et le processeur serait l'étudiant lui-même.

2.1.2 La mémoire

La mémoire est le deuxième composant en importance dans un ordinateur. Lorsque les programmes sont exécutés, ils sont d'abord chargés en mémoire afin d'être traités. De plus, lorsque vous éditez un texte, celui-ci est situé dans la mémoire au moment de l'édition. Bref, chaque tâche effectuée sur un ordinateur doit temporairement enregistrer des informations en mémoire.

Information

Il est indiqué *« temporairement »* et pour cause ! La mémoire se vide dès que la machine cesse d'être alimentée en courant. On dit que la mémoire est *volatile*. D'où l'importance de sauvegarder les fichiers sur disque.

Ce type de mémoire s'appelle la mémoire vive (par opposition à la mémoire morte) car son contenu peut être modifié à volonté. Les termes anglais utilisés sont *main memory* ou *RAM*.

Puisque chaque donnée utilisée et chaque programme exécuté se trouvent en mémoire, il est naturel de dire que plus nous en avons, mieux c'est ! Effectivement, si nous manquions de *RAM*, les performances de l'ordinateur diminueraient. En 2002, il n'est pas rare de voir des ordinateurs avec 512 Mo (méga-octets - unité de mesure d'espace mémoire) de *RAM*.

Continuons l'analogie. L'étudiant est le processeur, son pupitre de travail est sa mémoire. Lorsque l'étudiant fait des travaux, il place ses notes de cours (programmes) et ses travaux (données) sur sa table de travail. Plus il a d'espace sur sa table, plus il peut avoir de cartables ouverts en même temps.

2.1.3 Le disque dur

Le **disque dur** est l'endroit où les données et programmes non utilisés sont enregistrés. Contrairement à la *RAM*, les données sur disque sont enregistrées de manière permanente (à moins d'un bris matériel grave). Ceci est un réel avantage sur la *RAM*. Malheureusement, la technologie utilisée pour les disques durs (mécanique) ne permet pas un accès aussi rapide qu'à la *RAM* (électronique). C'est pourquoi les disques durs ne sont pas aussi dispendieux que la *RAM*. Les disques durs d'aujourd'hui (2002) se mesurent en termes de giga-octets.

Information

1 octet = 8 bits ;
1 Ko = 1024 octets ;
1 Mo = 1024 Ko ;
1 Go = 1024 Mo

Il existe un lien étroit entre la *RAM*, le disque dur et le processeur. Supposons, par exemple, que vous désiriez exécuter un programme de traitement de texte. Le programme est enregistré sur le disque dur. Le processeur charge le programme en mémoire vive et commence à l'exécuter. Une fois rédigé, le texte est sauvegardé sur le disque dur pour le conserver une fois l'ordinateur éteint.

Dans l'analogie, le disque dur est le tiroir du pupitre. Les cartables que l'étudiant n'utilise pas sont rangés dans le tiroir. Pour y accéder, l'étudiant doit d'abord sortir le cartable du tiroir et le placer sur sa table (charger en *RAM*).

Évidemment, un disque dur peut faire défaut et dans ce cas, toutes les données sont perdues (même le devoir du cours IFT1010 qu'il faut remettre le lendemain !). C'est pourquoi il existe des solutions de *stockage* amovible sur lesquelles on peut faire des copies de sûreté.

disquette Les disquettes sont comme des petits disques durs. Une disquette est un objet rectangulaire avec un disque magnétique à l'intérieur où les données sont enregistrées. Les disquette ont une moins grande capacité, sont très lentes et sont en voie de disparaître mais rien de mieux pour garder une copie d'un travail à remettre.

CD-R, CD-RW, DVD+-R, DVD+-... Les supports optiques, disques compacts inscriptibles et réinscriptibles et disques vidéos numériques (*digital video disks* ou dvd) de toutes sortes sont une alternative. Ils offrent plus de capacité que les disquettes et sont plus rapides. Cependant, même si tous les ordinateurs possèdent un lecteur de disquettes, on ne peut pas en dire autant pour un graveur de disques compacts.

> Un tel appareil est nécessaire seulement pour écrire sur un disque compact. En effet, on peut lire d'un disque compact (CD-ROM, CD-R, CD-RW) avec un lecteur de CDROM ordinaire.

Information

Unités de stockage usb Les unités de stockage usb sont de plus en plus populaires puisqu'elles permettent des transferts nettement plus rapides qu'une disquette, plus de capacité, utilise un connecteur usb que l'on retrouve maintenant sur presque toutes les machines et qu'elles peuvent être déplacées facilement. Les stockages usb peuven être séparés en plusieurs sous classes. Il y a les clés usb, qui sont très petite : et ont des capacités entre 64Mo et quelques gigaoctets. On peut aussi retrouver les autres types de stockage (ex. : disques durs, graveurs CD/DVD) avec une interface usb. Cela peut être pratique quand on veut pouvoir transférer de grandes quantité de donnés ou quand l'espace physique manque (ex. : ordinateur portable).

2.1.4 Clavier et souris

Le **clavier** et la **souris** font partie de ce qu'on appelle les **périphériques** d'entrée. Le mot périphérique signifie les appareils qui se trouvent autour de la machine centrale, c'est-à-dire autour du processeur et de la mémoire. À ce mot, on ajoute le mot entrée qui signifie que l'appareil sert à entrer des données dans la machine (*input peripheral*). Bien entendu, il existe aussi des *output peripherals* servant à sortir des données de la machine (ex. écran, imprimante).

Le clavier et la souris, en particulier, attendent des données venant de vous. C'est avec ces outils que vous pouvez communiquer avec l'ordinateur pour lui demander d'exécuter vos programmes et

vos tâches quotidiennes. Pour ceux qui sont vraiment débutants, le clavier est situé devant vous avec ces nombreuses touches. (Chaque touche a un symbole imprimé dessus - comme un clavier de dactylo). La souris, quant à elle se situe à ses côtés. C'est l'appareil qui a généralement deux ou trois boutons et qui se trouve sur un petit tapis.

Le clavier sert à entrer des informations textuelles pour la plupart du temps. Cependant, dans certains logiciels, des commandes peuvent être activées avec les touches du clavier. Par exemple, la combinaison CTRL-X CTRL-C vous permet de quitter Emacs (l'éditeur de texte). Dans la littérature informatique, vous trouverez souvent des symboles pour indiquer les touches. Chacun de ces symboles (ou sa traduction anglaise) est imprimé sur une touche du clavier. De plus, il n'est pas rare de voir des combinaisons telles que *« CTRL-X »*, qui signifie : appuyez sur CTRL, maintenez le enfoncé, appuyez sur X, relâchez le tout. Une autre manière d'écrire les combinaisons est d'indiquer la séquence de touches à enfoncer dans l'ordre. Par exemple, A, B, C signifie d'appuyer sur les touches A, B et C une après l'autre, mais sans les tenir enfoncées tout au long de l'opération.

Information

S'il vous arrivait d'appuyer sur 'Q' et de voir un 'A' apparaître, c'est que votre clavier est mal configuré. En effet, pour fonctionner, le clavier doit être configuré pour que les informations qu'il envoie à l'ordinateur soit traitées correctement.

Information

Dans plusieurs logiciels, dont la console Linux, vous voyez un petit carré qui clignote. Ce petit carré est appelé un curseur (*carret*) et il sert à vous indiquer où vous êtes rendu dans la ligne.

La souris, quant à elle, sert à contrôler le **pointeur**. En déplaçant la souris, vous faites avancer le pointeur. Cette action s'appelle **pointer** car vous pointez vers un objet avant de le sélectionner. Une fois le pointeur déplacé, vous pouvez **cliquer** ou **double-cliquer**, ce qui accomplira une action différente selon le cas.

Voici la terminologie de base pour la souris :

pointeur flèche sur l'écran suivant le mouvement de la souris ;

pointer déplacer le curseur à un endroit précis de l'écran ;

enfoncer appuyer sur un bouton de la souris et le garder enfoncé ;

cliquer pointer à un endroit précis, appuyer sur un bouton de la souris et le relâcher rapidement ;

double-cliquer cliquer deux fois rapidement sans déplacer la souris ;

traîner enfoncer en déplaçant la souris.

2.2 Le traitement de données

Passons maintenant à l'aspect logiciel (*software*) de l'informatique.

2.2.1 Qu'est-ce qu'un programme ?

Un programme informatique n'est rien de plus qu'une série d'instructions que le processeur doit exécuter. Les instructions peuvent servir à lire des données sur le disque, demander une entrée au

clavier ou afficher des dessins sur l'écran. Lorsque ces instructions sont organisées dans une séquence logique dans un but particulier (calculer, jouer, etc), nous appelons cette séquence un programme informatique. On dit alors que le programme roule (*run*) sur la machine. Comme vous le savez déjà, le processeur est le cerveau et c'est lui qui s'occupe d'exécuter toutes les tâches demandées dans le programme. Les programmes sont codés (programmés) par des programmeurs dans différents langages (les langages de programmation) qui sont compris par la machine et par l'homme.

2.2.2 Qu'est-ce qu'un fichier ?

Comme l'informatique est la science du traitement de l'information, il est normal de discuter de la représentation de celles-ci. La plupart des informations qui se retrouvent sur un disque dur y sont sous la forme d'un fichier. Un fichier est une séquence d'information sur le disque à laquelle on donne un nom. Par exemple, certains fichiers contiennent des images, d'autres des textes. Un programme peut lire un fichier sur le disque dur et effectuer un traitement avec l'information qu'il contient. Par exemple, *GIMP* (ou Photoshop) peut ainsi lire une image dans un fichier sur le disque et l'afficher à l'écran. Lorsque vous travaillez sur un devoir de programmation (IFT1010), vous enregistrez votre devoir dans un fichier qui contient du texte, dans ce cas, des instructions.

Certains systèmes distinguent deux types de fichiers, les fichiers texte et les fichiers binaires. Les fichiers texte sont ceux qui peuvent être lus par un humain et qui ne contiennent que des caractères affichables qui sont décrits dans des standards comme ASCII. Les fichiers binaires contiennent quant à eux des informations destinées à être lues par un programme et non par un être humain. Ce type de fichier est donc incompréhensible pour l'être humain, mais contient tout de même des informations pertinentes.

Un autre type de fichier qu'il est important de connaître est le fichier exécutable. Le fichier exécutable est celui qui contient des instructions à être exécutées par la machine. Sur Linux, tout fichier peut être déclaré exécutable. Sous Windows, les fichiers exécutables sont ceux qui portent les extensions EXE, COM et VBS (parmi les plus connues).

Les fichiers exécutables contiennent des instructions qui seront exécutées par votre machine. Ainsi, il peut arriver que certains d'entre eux contiennent des instructions servant à endommager d'une manière ou d'une autre votre ordinateur ; ce sont les *« virus »*. Pensons simplement au fameux virus *« I love you »* qui n'était rien de plus qu'un fichier exécutable de type VBS.

Attention

Vous verrez dans vos cours que les différents types de fichiers ne sont qu'un outil de compréhension et qu'en fait, à un niveau plus bas, un fichier exécutable reste un fichier et rien le distingue d'un fichier non exécutable.

Information

Un fichier est identifié par son nom. Le nom choisi pour un fichier est important car c'est avec celui-ci qu'on identifie rapidement le contenu du fichier. Quel nom est le plus approprié pour la photo de ma copine : `lara.jpg` ou bien `photo.jpg` ? Comme vous venez de le constater, certains systèmes (Windows) séparent le nom d'un fichier en deux parties : le nom et l'extension. Vous comprenez maintenant qu'il est important de choisir un bon nom. Pour l'extension, elle sert à identifier le type du fichier. Par exemple, .jpg signifie 'Image JPEG', .exe signifie programme exécutable, etc. Les

extensions servent à Windows pour savoir quel programme doit manipuler le fichier. Sous UNIX, l'extension n'est pas aussi importante. En fait, la plupart des programmes UNIX ignorent l'extension (voir la section 3.4.1 pour plus de détails sur les noms de fichiers sous UNIX).

2.2.3 Les répertoires

Un répertoire est un fichier spécial ; les informations qu'il contient sont en fait d'autres fichiers. En effet, un répertoire est comme une chemise dans laquelle on range des feuilles. Plusieurs fichiers sont rangés dans le répertoire ce qui nous permet de classer (et retrouver) rapidement nos fichiers. Dans la plupart des systèmes, les répertoires sont organisés dans un arbre avec un lien hiérarchique. Voici deux exemples concrets.

Sous Linux, *« / »* est le répertoire racine, la racine de l'arbre. Sous cette racine, on trouve d'autres répertoires comme *« /home »* qui contient les fichiers des usagers, *« /bin »* qui contient les fichiers des programmes de base, *« /doc »* qui contient les fichiers d'aide et de documentation, etc.

Sous Windows, la racine est souvent `C :\` qui signifie, disque dur C, racine. Dans l'arbre, on retrouve souvent `C :\WINDOWS` qui contient les fichiers relatifs au système et peut-être `C :\JEUX` qui contient vos jeux préférés. Les versions récentes de Windows contiennent également `C :\Program Files` où les programmes sont enregistrés et `My Documents` qui fait office de *home* sous Windows.

Information

> Comme vous pouvez le constater, une des bases de l'informatique est de comprendre que malgré les différents systèmes qui existent, les concepts restent les mêmes. Ainsi, les noms diffèrent, le *look* change, mais les concepts fondamentaux ne changent pas beaucoup. Une fois que vous avez saisi les concepts, vous pouvez apprendre n'importe quel système facilement en très peu de temps.

En anglais, les termes *folder* et *directory* sont utilisés à la place de répertoire.

2.2.4 Qu'est-ce qu'un logiciel ?

Nous arrivons enfin au concept qui vous est le plus familier, mais pourtant qui est un des plus flous, le logiciel. Qu'est-ce qu'un logiciel ? Peut-être vos études universitaires en informatique sont-elles motivées par l'objectif d'écrire des logiciels ? Le logiciel est en quelque sorte un tas de programmes et de fichiers qui ensemble, offrent des outils à des utilisateurs. Il existe plusieurs catégories de logiciels : systèmes d'exploitation, jeux, bureautique, etc. Il existe une panoplie de concepts reliés aux logiciels et leur ingénierie. C'est pourquoi vous êtes encouragé à consulter les professeurs des cours de génie logiciel (IFT2251 par exemple) pour avoir un regard nouveau sur la vérité à propos des logiciels.

2.2.5 Qu'est-ce qu'un système d'exploitation ?

Les logiciels qui sont sans aucun doute les plus importants sont les systèmes d'exploitation. Un système d'exploitation est un logiciel qui s'occupe de gérer les ressources de l'ordinateur et qui offre à l'usager une interface générale pour communiquer avec l'ordinateur. Vous ne voudriez pas communiquer avec la machine dans son langage en tout temps, croyez-moi. Il est plus utile de se servir des fonctionnalités du système d'exploitation. Pour mieux comprendre ce qu'est vraiment le

système d'exploitation, il est bon de visionner la machine comme une structure avec des couches. La première couche en bas serait le matériel. Juste au dessus est situé le système d'exploitation qui s'occupe de communiquer avec le matériel à votre place. Puis viennent les applications ordinaires comme les jeux, les logiciels de bureautique, etc.

En anglais, le système d'exploitation s'appelle *operating system* ou OS. Voici des exemples de systèmes d'exploitation connus : DOS, Windows, Linux, Solaris, MacOS. Dans le cours IFT2240, vous verrez comment fonctionne et comment est programmé un système d'exploitation.

2.3 La communication entre ordinateurs

2.3.1 Les réseaux

Un réseau, dit simplement, c'est un ensemble d'ordinateurs qui communiquent entre eux. Comme toute forme de communication, il y a certains éléments qui sont nécessaires pour que deux ordinateurs puissent se parler. Pour que deux humains puissent se parler, il faut un milieu que l'information puisse traverser (l'air) et une façon pour encoder et décoder l'information qu'on veut envoyer (le langage). Le milieu physique pour transmettre l'information dans le cas d'un réseau est très souvent un fil qui relie les deux ordinateurs, bien que des réseaux sans fil existent. Les langages qu'utilisent les ordinateurs pour communiquer sont nommés **protocoles de communication**. Tout comme les langages chez les humains, il existe plusieurs protocoles de communication (IPX, TCP/IP, NetBEUI, ATM, Myrinet en sont des exemples). Les nuances entre ces protocoles sont sans importance dans le cadre de ce texte. Sachez seulement qu'il en existe plusieurs, que certains protocoles sont meilleurs pour certaines tâches et que deux ordinateurs qui n'utilisent pas le même protocole ne pourront tout simplement pas échanger de données.

Les concepts reliés aux protocoles de communication sont exposés dans le cours IFT3320, *« Téléinformatique »*.

2.3.2 L'Internet

En ayant une compréhension de ce qu'est un réseau, il est beaucoup plus facile de définir ce qu'est l'Internet. L'Internet, c'est un gros réseau mondial. Ce réseau est tellement gros qu'on est obligé de le subdiviser de manière hiérarchique. Donc en fait, l'Internet est plutôt un ensemble de réseaux qui sont reliés en réseau. Ces sous-réseaux peuvent eux-mêmes être des réseaux de réseaux. Les machines qui composent l'Internet sont extrêmement variées. Elles varient entre l'ordinateur personnel sur lequel vous vous connectez à l'Université et de gigantesques *routeurs* qui gèrent le trafic. À moins d'un isolement volontaire, n'importe quel ordinateur sur l'Internet peut communiquer avec n'importe quel autre.

Il y a cependant plusieurs millions d'ordinateurs sur l'Internet qui se parlent tous et qui n'ont pas nécessairement les mêmes buts : Monsieur Untel qui est à la maison veut vérifier les conditions météorologiques sur le Web tandis que Madame Untel qui est au travail veut transférer un document à son collègue et leur fils qui est à l'école veut vérifier son courrier électronique. Toutes ces applications n'utilisent pas nécessairement le même protocole de communication, mais ces derniers reposent tous sur une base commune. On peut donc dire qu'il existe un seul protocole normalisé sur l'Internet sur lequel peuvent se rajouter d'autres protocoles à usages variés et ce, de manière transparente.

Le cours IFT3320 explique en détail les technologies de l'Internet.

2.3.3 Le World Wide Web

Le Web est un des multiples services offerts sur l'Internet. Ce service est d'ailleurs tellement populaire que plusieurs font l'erreur de confondre l'Internet et le Web. Le Web fut conçu pour permettre la présentation de documents multimédia. Ainsi, en plus de son contenu textuel, la page Web typique nous présente du graphisme et des extraits sonores et visuels. Le grand avantage du *« WWW »* réside dans le fait que l'utilisateur n'a qu'à choisir à travers le document (communément appelé page Web) un lien hypertexte pour que de nouvelles informations lui parviennent. En fait, l'utilisateur n'a pas à se soucier de la façon dont les documents sont retrouvés et acheminés jusqu'à son terminal. Ce sont donc des millions de pages traitant de tous les sujets possibles qui deviennent accessibles à tous.

Le Web peut être vu comme une immense bibliothèque virtuelle internationale : des millions d'étagères contenant des millions de livres (sites Web). Les livres sont classés pêle-mêle : on ne peut pas parcourir tous les sites Web dans un ordre défini. Pour trouver ce que l'on cherche, on utilise des **engins de recherche**, des sites Web qui indexent le Web, auxquels on indique ce que l'on cherche et qui nous suggèrent des sites à consulter. Une bonne partie des sites sont créés et entretenus par des compagnies, qui peuvent s'en servir comme d'une brochure publicitaire, d'un formulaire, d'un catalogue, etc.

De la même manière qu'un livre, un site est un ensemble de pages ; toutefois contrairement à un livre, les pages d'un site Web ne sont pas forcément en ordre.

Pour naviguer sur le Web, il vous faudra utiliser un *« fureteur »*. Le premier fureteur graphique vit le jour vers 1990 et s'appelait *« WorlDwidEweb »*. Depuis, des dizaines de navigateurs furent lancés. L'un des plus populaires actuellement est sans aucun doute **Firefox**. Voyez la section 4.1.1 pour plus d'informations concernant les navigateurs.

URL

Histoire courte du World Wide Web
`http ://www.w3.org/People/Berners-Lee/ShortHistory`

2.3.4 Le courrier électronique

La messagerie électronique (envoi de courriels ou *email* : *electronic mail* pour les intimes) est un autre service offert sur l'Internet. Tout comme chaque individu possède sa propre adresse postale, chaque usager dispose aussi d'une adresse qui lui est propre. Ces adresses ont le format suivant :

```
usager@système.domaine
```

où un domaine est une liste de sous-domaines séparés par un point ('.'), et où usager est le nom de l'usager sur son système. Habituellement un domaine est composé de trois sous-domaines, mais il peut y en avoir plus ou moins. Le format d'adressage ressemble souvent à ceci :

```
usager@RéseauInterne.organisation.pays
```

Avec les différents sous-domaines, un message trouvera la personne à qui il est destiné. Prenons un exemple d'adresse et examinons le cheminement que va suivre le message. Supposons qu'un message est envoyé à :

```
laplante@IRO.UMontreal.Ca
```

Lorsque la machine s'occupant de gérer le courrier électronique (appelée *« mail server »*) est prête à envoyer le message, elle commence par vérifier l'adresse qui est fournie. Elle commence par la droite avec le 'Ca' : ce sous-domaine représente le Canada. Ce sous-domaine étant valide, la machine passe au sous-domaine suivant : 'UMontreal' représente l'Université de Montréal et 'IRO' représente un des réseaux de l'Université. Enfin, *« laplante »* représente un usager sur ce réseau (la personne à qui est adressé le message).

L'adresse étant bonne, le message est envoyé. Si la machine avait trouvé un sous-domaine invalide, un message d'erreur serait retourné à l'expéditeur avec une copie du message. Cet exemple comporte les trois sous-domaines habituels. La structure d'une adresse dépend beaucoup de la façon dont le réseau recevant le message a été implémenté. Il est à noter que les noms des sous-domaines peuvent être donnés en minuscules ou en majuscules pour tous les types d'adressage. Notez aussi que les noms de sous-domaines (et d'usagers) n'ont pas d'accents.

Chapitre 3

Les bases de UNIX

Ce chapitre vise à enseigner les bases fondamentales d'un système UNIX. L'étude de ce chapitre devrait suffire à vous donner une compréhension globale d'un système UNIX et à vous permettre de vous y débrouiller et d'y travailler. Comme vous le verrez bientôt, UNIX est un système complexe qui ne peut être résumé en quelques pages. L'information contenue dans ce chapitre devrait toutefois être suffisante pour vous permettre de vous lancer dans l'apprentissage de cet environnement de travail.

3.1 Philosophie

Lorsque nous travaillons avec UNIX, il faut toujours garder en tête certaines notions qui ont inspiré ses créateurs. Ces notions se reflètent tant dans chaque programme UNIX que dans leur agencement.

- Chaque programme ne fait qu'une seule fonction simple et il la fait bien.
- La sortie d'un programme sert éventuellement d'entrée à un autre et ainsi de suite. Cela servira à construire des outils complexes à partir de petits programmes simples.
- Lorsque nous ne trouvons pas le programme adéquat, il ne faut pas hésiter à l'écrire ; la bibliothèque va s'enrichir. (Il ne faut toutefois pas chercher à *« réinventer la roue »* - des milliers de programmes remplissant les tâches les plus variées sont déjà disponibles).

UNIX est un système d'exploitation multitâche, multiprocesseur et multiutilisateur. Il offre une multitude de petits outils simples mais très efficaces.

C'est aussi un système peu onéreux. Plusieurs outils et même des systèmes d'exploitations sont gratuits ; plusieurs distributions du système Linux, chacune avec sa sélection de milliers d'outils, sont disponibles gratuitement.

Le mouvement du *« logiciel libre »* (*Free Software*) est très actif et encourage les auteurs de logiciels à distribuer leur code source afin de permettre à d'autres auteurs de le réutiliser. Cette philosophie permet aux logiciels d'évoluer naturellement.

Free Software Foundation
`http ://www.fsf.org`

URL

3.2 Concepts de base

3.2.1 Majuscules vs. minuscules

Contrairement à d'autres systèmes d'exploitation (comme Windows), UNIX fait la différence entre les **MAJUSCULES** et les **minuscules**. *« DIRO »*, *« Diro »* et *« diro »* sont trois mots différents. Ceci est valide entre autres pour les noms de fichiers et des commandes.

3.2.2 Le terminal

La majeure partie de l'interaction entre un système UNIX et ses usagers se fait via la ligne de commande (*prompt*). Même si les versions récentes de systèmes comme Linux permettent de travailler uniquement via l'interface graphique (comme sous Windows ou MacOS), la ligne de commande permet un meilleur contrôle du système et permet de combiner des programmes d'une manière autrement impossible. Il est donc **essentiel** d'apprendre à se servir d'un terminal.

Souvent, quand vous vous branchez à distance (voir section 6.3) tout ce que vous avez est un terminal — vous n'avez pas d'interface graphique pour travailler. Quand vous vous branchez sur place (dans un labo), vous pouvez ouvrir un terminal en cliquant sur l'icône le représentant (voir la figure 3.1).

FIG. 3.1 – La console

L'interaction avec un terminal est assez simple : vous tapez une commande suivie de la touche `[Entrée]` et le système exécute la tâche décrite dans votre commande. Ceux qui ont connu DOS sont déjà habitués au principe. La syntaxe générale d'une commande est décrite à la section 3.5.1.

3.3 Les usagers

UNIX est un système d'exploitation multiutilisateurs sécuritaire. Pour travailler sur une machine UNIX l'utilisateur doit y posséder un compte.

Avant toute session de travail sur le système, l'utilisateur est authentifié, et, selon ses droits d'accès sur le système, il pourra accéder à certaines ressources. À chaque fois que vous créez un fichier ou que vous lancez un programme, le fichier ou l'exécution du programme sont identifiés comme vous appartenant. Cette appartenance permet au système d'appliquer les restrictions qui protègent des autres usagers le contenu de vos fichiers et l'exécution de vos programmes.

Information

Sur UNIX, chaque fichier et chaque exécution d'un programme est associée à un usager, son *« propriétaire »*.

L'utilisateur doit lire les politiques d'utilisation de son compte (lire la section 5.1).

3.3.1 Attributs d'un compte

Un compte comporte :

1. un nom d'usager qui sert à identifier l'utilisateur ;
2. un mot de passe qui sert à authentifier l'utilisateur ;
3. un répertoire de travail contenant les données personnelles de l'utilisateur ;
4. une adresse de courrier électronique ;
5. un site Web personnel.

Nom d'usager

Le nom d'usager, ou *login*, est le nom de l'utilisateur sur le système. Il vous est attribué lors de votre inscription et sert à vous identifier auprès du système. Il n'est pas possible de modifier son nom d'usager. Un nom d'usager compte généralement huit lettres ou moins, et est souvent composé à partir du nom réel de l'usager.

Mot de passe

Le mot de passe sert à vous authentifier auprès du système. Il doit rester confidentiel pour éviter qu'un autre utilisateur puisse se faire passer pour vous. Changez régulièrement votre mot de passe à l'aide de la commande `chmp`. Les mots de passe trop simples, ou basés sur un mot du dictionnaire seront refusés.

Voir la section 1.5 pour des explications concernant le changement de votre mot de passe.

Répertoire de travail

Un répertoire de travail personnel est donné à chaque utilisateur. Tous les fichiers personnels d'un utilisateur sont contenus dans son répertoire de travail.

L'utilisateur est responsable du contenu de ce répertoire de travail. Une limite d'espace est généralement imposée aux usagers pour éviter les excès.

Par défaut, les fichiers de votre répertoire de travail ne sont pas accessibles par les autres utilisateurs. Cependant, il est possible de modifier les droits d'accès des fichiers pour les partager avec les autres utilisateurs (voir la section 3.4.3 pour tous les détails).

Courrier

Un compte de courrier électronique, aussi appelé *« courriel »* ou *email*, est fourni avec votre compte. Votre adresse est du format `votrelogin@iro.umontreal.ca`.

Voyez la section 4.1.2 pour savoir comment utiliser la messagerie électronique.

Site Web personnel

Une page web personnelle est fournie avec votre compte. Si vous êtes familier avec les techniques d'édition de site Web (principalement le langage HTML, dont l'étude dépasse le cadre de ce document) vous pouvez modifier cette page et y mettre ce qui vous semble pertinent.

Les fichiers de votre page web se trouvent dans le sous-répertoire `/home/www-etud/usagers/login/HTML`. Vous devez ajuster les droits d'accès pour permettre aux utilisateurs hors du système d'accéder à ces ressources (voir la section 3.4.3).

3.3.2 Les groupes d'usagers

Les usagers d'un système UNIX sont distribués en groupes qui facilitent l'administration en permettant d'appliquer des restrictions ou permissions à tout un groupe prédéfini d'usagers. Un usager fait partie d'au moins un groupe.

Les groupes sont assez peu utilisés par les usagers qui ne sont pas des administrateurs. Au DIRO, tous les étudiants du baccalauréat font partie du groupe *« etud »*. Vous n'aurez pas à vous en soucier beaucoup.

3.3.3 Le superusager

Chaque système UNIX comporte un et un seul usager qui fait la loi. Cet usager se nomme *« root »* ; on l'appelle le superusager (*superuser* en anglais).

Le superusager n'est limité par aucune des règles de UNIX. Il peut lire les fichiers de tout le monde, interrompre tout programme, etc. Grâce à ces permissions, il peut exécuter des tâches administratives que personne d'autre ne peut faire.

3.4 Le système de fichiers

Dans cette section, nous décrirons les notions élémentaires du système de fichiers utilisé sous UNIX. Nous y étudierons la nomenclature et la structure des objets ainsi que leurs droits d'accès.

Vous viendrez à réaliser que le système UNIX est très orienté vers le concept de fichiers. Contrairement à d'autres systèmes d'exploitation populaires où certaines données sont enregistrées dans un endroit obscur connu seul du système (comme la base de registres de Windows, par exemple), en UNIX tout est fichier.

3.4.1 Les types de fichiers sous UNIX

Il existe plusieurs types de fichiers sous UNIX.

Fichier Objet contenant des données (texte, image, etc.) ou formant un programme exécutable ;

Répertoire Objet contenant d'autres fichiers (de n'importe quel type) ;

Périphérique Objet correspondant à une entité physique à laquelle l'ordinateur est relié, c'est-à-dire une imprimante, l'écran, un modem, un disque dur, etc. Ces fichiers *« virtuels »* ne sont manipulés que par le système lui-même et vous n'aurez pas à vous en servir.

Lien *« symbolique »* ou *« physique »*. Objet pointant sur un autre. Ceci permet de simuler l'existence de plusieurs copies d'un fichier alors que physiquement, il n'y en qu'une. Les liens symboliques peuvent remplir les mêmes usages que les *« raccourcis »* de Windows ou les *« alias »* de MacOS.

Les deux seuls types de fichiers dont il est essentiel de comprendre l'usage sont les fichiers simples et les répertoires. Les deux autres sont plutôt utiles aux utilisateurs avancés, en particulier les fichiers de périphériques. Il existe aussi d'autres types de fichiers, dont nous ne parlerons pas dans ce manuel.

Identificateur

Chaque objet contient un identificateur propre (son nom). Le nom du fichier peut être n'importe quelle chaîne de caractères, d'une longueur maximale de 255. Il peut contenir n'importe quels caractères, sauf la barre oblique « / », parce qu'elle sert à délimiter les répertoires (voir plus loin).

Tout fichier dont le nom commence par un point (« . ») est un fichier caché et ne sera pas immédiatement visible lorsque nous demandons la liste des fichiers d'un répertoire.

Certaines précautions doivent être prises si nous voulons attribuer des noms contenant des caractères spéciaux (tels '?', '*', '&', l'espace, etc) : ils sont admis, comme les autres caractères, mais rendent la manipulation des noms de fichiers à la ligne de commande plus pénible. En général, il est suffisant de mettre le nom entre apostrophes, mais nous déconseillons leur usage.

Il n'y a pas de restriction sur la structure de l'identificateur. Les identificateurs suivant sont valides :

- `fichier`
- `fichier.txt`
- `fichier-txt`
- `mon.fichier.txt`
- `mon fichier texte`
- `mon$fichier,texte`

Notez que contrairement à Windows, l'extension ne détermine pas de manière certaine le type de fichier et elle n'est pas nécessaire, si ce n'est pour aider l'usager à se souvenir du type de contenu du fichier.

Windows

3.4.2 Accès à un fichier

Une fois un fichier créé, il existe de manière permanente sur le disque, c'est à dire que tant que l'usager ne demande pas explicitement de l'effacer, il demeure accessible au même endroit sur le disque dur.

Le système de fichiers de UNIX peut être vu comme un arbre : il comporte un point de base, la racine du système. La racine est désignée par la barre oblique seule : « / ». La racine est un répertoire contenant plusieurs sous-répertoires, qui eux-mêmes en contiennent d'autres, etc. La racine est le seul répertoire n'ayant pas de parent. La figure 3.2 illustre la hiérarchie du système de fichiers.

Contrairement à Windows, la système de fichiers de UNIX n'a qu'une seule base. Windows comprend plusieurs « *unités* » parallèles (C, D, etc.) ; sous UNIX tout le système de fichiers fait partie du même arbre

Windows

Un chemin d'accès (*path*) permet de localiser un objet sur le système de fichier. Comme des directives routières, le chemin d'accès indique la séquence de répertoires par laquelle passer pour

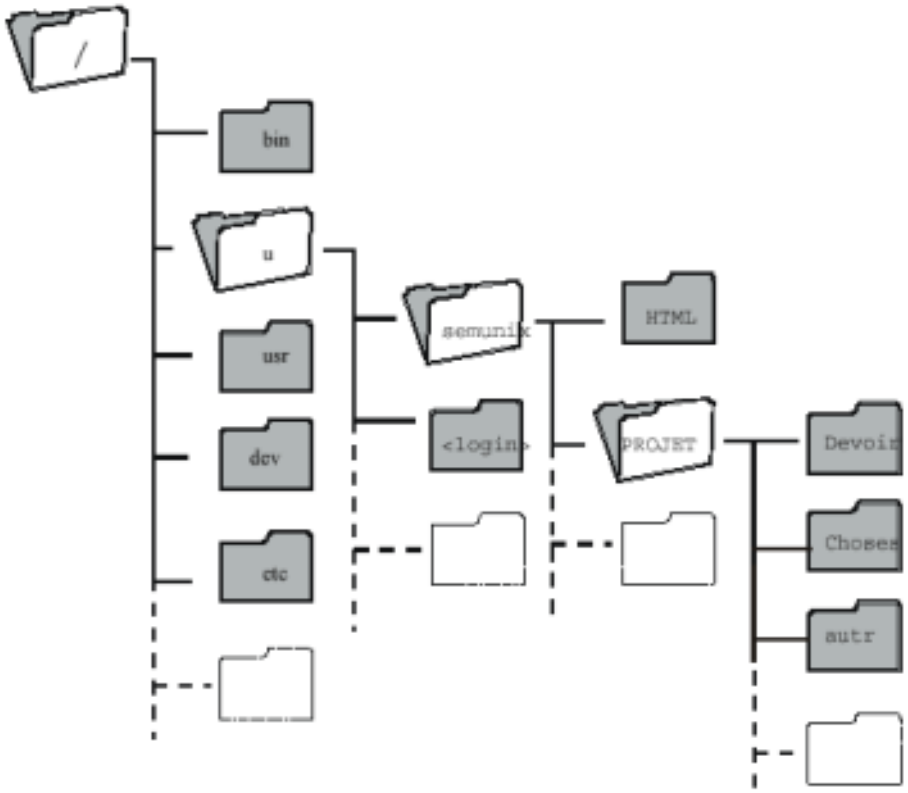


FIG. 3.2 – Hiérarchie du système de fichiers

se rendre d'un répertoire à un autre. On sépare les noms des répertoires successifs par des barres obliques.

Il existe deux manières de nommer un fichier : **absolue** et **relative**.

Chemin absolu

Quand on donne le chemin d'accès absolu d'un fichier dans l'arborescence, on donne sa position par rapport à la racine. Partant de la racine, on peut suivre le chemin donné et aboutir au fichier. Comme le chemin part de la racine, il commence par une barre oblique « / ».

Par exemple, dans la figure 3.2, le répertoire `PROJET` est désigné par le chemin absolu `/u/semunix/PROJET`. D'autres exemples :

```
/u/semunix/semunix.pdf
/bin/ls
/usr/local
```

Vous pouvez remarquer que les chemins absolus commencent tous par une barre oblique.

Chemin relatif

Il aussi possible de localiser un fichier par rapport à un autre. On décrit alors le chemin non pas à partir de la racine mais à partir d'un autre répertoire.

Il faut parfois remonter dans la hiérarchie (par exemple, dans la figure 3.2, pour se rendre dans `Devoir` à partir de `HTML`, il faut remonter d'un niveau puis aller dans `PROJET/Devoir`). Pour désigner le parent d'un répertoire, on utilise la notation `..` :

- Le répertoire `..` représente le chemin relatif du parent du répertoire courant (dans lequel on se trouve).
- Le répertoire `.` représente le chemin relatif du répertoire courant.

Le chemin d'accès du répertoire `Devoir` relatif au répertoire `HTML` est donc `../PROJET/Devoir`. Le répertoire `semunix`, lui, est simplement désigné par `..`, puisqu'il est le parent de `HTML`.

Autres exemples de chemins relatifs (les deux derniers sont équivalents) :

```
../texte.txt
cours/texte.txt
./cours/texte.txt
```

3.4.3 Les droits d'accès

UNIX, étant un système multiutilisateurs, permet de définir des droits d'accès sur les composantes du système de fichier (fichier, répertoire, périphérique, ...). Ainsi, il est possible de conserver pour soi certains fichiers personnels et de partager d'autres fichiers à des groupes d'utilisateurs.

Les fichiers et répertoires se trouvant dans votre répertoire principal sont bien à vous. Cependant, le système UNIX est un système qui se dit *« ouvert »* : il permet de partager les ressources. Cela laisse donc la possibilité à toute personne ayant accès au réseau d'utiliser des fichiers qui se trouvent dans votre compte personnel. Dans une entreprise où plusieurs personnes travaillent sur le même projet, ceci peut s'avérer très pratique. Un peu moins pratique cependant dans le cas d'une université où près d'une centaine d'élèves auront à faire les mêmes travaux. Certains chercheront peut-être à profiter de cette ouverture. C'est pourquoi il existe différents systèmes de protection.

Comme premier exemple d'inspection des permissions d'accès d'un fichier, vous allez inspecter celles de votre répertoire personnel. Tapez la commande suivante :

```
$ ls -ld ~
```

(Voir la section 3.5.4 pour plus de détails sur la commande `ls`)

Vous obtenez alors quelque chose de similaire à ce qui suit :

```
drwxr-xr-x   58 semunix  etud         5120 Jul 26 12:19 /u/semunix
```

La première colonne à gauche, composée de lettres et de tirets, donne les permissions du fichier dont le nom se trouve à droite complètement. Dans l'exemple qui précède, ce fichier est votre répertoire de travail personnel. La colonne de gauche indique donc qui possède quelles droit sur votre répertoire de travail.

Pour décoder les permissions, il faut d'abord décomposer l'information comme ceci :

[d] [rwx] [r-x] [r-x].

Le premier bloc, composé d'une seule lettre (dans ce cas **d**), indique type du fichier. Les types de fichiers fréquents sont :

- **-** **file** Fichier normal.
- **d** **directory** Répertoire.
- **l** **link** Lien.

Le **d** indique donc qu'il s'agit d'un répertoire.

Suivent les trois triplets. Chacun est composé des lettres *« rwx »*, toujours dans cet ordre ; certaines lettres sont toutefois masquées par un tiret. Dans chaque triplet, chaque lettre indique un type particulier de permission.

- **r** **read** Permission de lire.
- **w** **write** Permission d'écrire.
- **x** **execute** Permission d'exécuter.

Si la lettre est visible, la permission est donnée. Si elle est cachée par un tiret, est n'est pas donnée. Les trois blocs donnent les droits d'accès pour, dans l'ordre, le propriétaire du fichier (*owner*), les membres du groupe (*group*) et les autres (*others*).

Dans l'exemple qui précède, les permissions sont `rwxr-xr-x`, voyons ce que cela signifie :

1. Le premier bloc indique les permissions du propriétaire (vous, dans ce cas). Il contient **rwx** ; aucune lettre n'est masquée, ce qui signifie que vous avez tous les droits : lecture, modification, et exécution.
2. Le second bloc indique les permissions des usagers dans le même groupe (les autres étudiants du bacc, dans ce cas). Il contient **r-x** ; la lettre **w** est masquée, ce qui signifie que les autres étudiants n'ont pas de droit de modification. Les lettres **r** et **x** sont toutefois visibles, donc ils ont tout de même le droit de lecture et d'exécution.
3. Le troisième bloc indique les permissions des autres (tous les usagers qui ne sont pas au bacc, incluant les *« visiteurs »*, comme ceux qui consultent votre page Web par exemple). Il contient la même chose que le bloc précédent, ce qui signifie que les autres ont les mêmes droits que ceux du groupe, c'est-à-dire le droit de lecture et le droit d'exécution, mais pas celui de modification.

Voyons maintenant le sens de ces permissions. Le sens des termes *« droit d'écrire, droit d'exécuter »* diffère selon qu'ils s'appliquent à un fichier simple ou à un répertoire.

fichier

- **r** **read** Permet la lecture du contenu du fichier.
- **w** **write** Permet l'écriture, c'est-à-dire la modification, du fichier.
- **x** **execute** Permet d'exécuter le fichier.

répertoire

- **r** **read** Permet de lister le contenu du répertoire.
- **w** **write** Permet de créer, d'effacer ou de renommer les fichiers du répertoire.

x **execute** Permet d'entrer dans le répertoire.

Le propriétaire d'un fichier peut en changer les permissions avec la commande `chmod` ; voir la section 3.5.10.

3.5 Les commandes de base

3.5.1 La syntaxe générale d'une commande

Il a été dit plus tôt que pour apprécier pleinement la puissance de UNIX il faut savoir se servir de la ligne de commande. Dans le terminal, on tape des commandes : ce sont des instructions qui décrivent à l'ordinateur les tâches que nous voulons qu'il accomplisse.

Les commandes ont la syntaxe générale suivante.

```
$ commande [options] [paramètres]
```

`$`	Le `$` est l'invite (*prompt*) de votre interpréteur de commande, et il ne faut pas le taper. Il indique que le système attend une commande.
`commande`	Le nom de la commande.
`options`	La plupart des commandes possèdent des options facultatives qui en modifient le comportement.
`paramètres`	Les commandes prennent généralement des paramètres. Les paramètres sont les données d'entrées à la commande. Souvent, les paramètres sont des noms de fichier.

commande

Le nom de la commande. Il décrit l'action que nous voulons que l'ordinateur exécute. Le système offre un assez grand nombre de commandes (des milliers).

options

Chaque commande offre des options qui peuvent modifier son comportement. Les options sont souvent nommées par une seule lettre et précédées d'un tiret (« - »).

Rares sont les commandes qui nécessitent absolument des options : la plupart peuvent être lancées sans en spécifier.

La plupart du temps, l'ordre dans lequel on donne les options n'a pas d'importance. Aussi, la plupart des commandes permettent de regrouper les options : `ls -l -a` est équivalent à `ls -la`.

Vous verrez plusieurs exemples d'options dans les sections décrivant les commandes courantes.

Une option très courante est `--help` (avec deux tirets) ; cette option indique à la commande de ne pas s'exécuter et de décrire toutes les options possibles.

paramètres

Plusieurs commandes s'appliquent à quelque chose (souvent un fichier). Par exemple, la commande pour supprimer un fichier doit savoir quel fichier supprimer : le nom de ce fichier est donné en paramètre.

Voyons maintenant les commandes les plus courantes. Pour devenir efficace sous UNIX vous devriez connaître ces commandes par cœur.

Dans les exemples d'interaction avec le terminal qui suivent, ce qui est en **gras** est ce que l'usager (vous) doit taper. Ce qui n'est pas est ce qui est répondu par le système. Le `$` représente l'invite du terminal (voir plus haut) : vous ne devez pas le taper.

```
$ vous tapez ceci
Le syst\'eme vous r\'epond ceci
```

3.5.2 pwd

La commande `pwd` (*Print Working Directory*) affiche le nom du répertoire courant. Si vous la tapez dans un terminal que vous venez juste d'ouvrir, la commande affichera le nom de votre répertoire de travail personnel : quand vous démarrez le terminal, vous vous vous trouvez toujours dans votre répertoire personnel.

```
$ pwd
/u/semunix/
```

3.5.3 cd

La commande `cd` (*Change Directory*) sert à changer de répertoire courant. En d'autres mots, elle sert à se déplacer dans le système de fichiers.

Nous avons vu que quand vous ouvrez une console, vous vous trouvez dans votre répertoire de travail personnel. Nous avons aussi vu que votre répertoire de travail personnel contient un sous-répertoire appelé `PROJET`. Nous allons maintenant nous y déplacer :

```
$ cd PROJET
$ pwd
/u/semunix/PROJET
```

Après nous être déplacés dans le sous-répertoire `PROJET`, le commande `pwd` nous indique que nous avons changé de répertoire courant.

Comme nous l'avons vu à la section 3.4.2, il existe deux façons de nommer un fichier : de manière absolue ou relative. L'exemple précédent utilise la manière relative : à partir de votre répertoire de travail, on va dans `PROJET`. Comme le chemin d'accès absolu de votre répertoire de travail ressemble à `/u/semunix`, le chemin d'accès relatif de son sous-répertoire `PROJET` est `/u/semunix/PROJET`. Cela

signifie que pour aller dans ce répertoire, vous pouvez taper `cd /u/semunix/PROJET`, peu importe là où vous vous trouvez.

Nous avons aussi vu que le chemin relatif pour le répertoire parent est `..` ce qui signifie qu'une fois dans le répertoire `PROJET`, la commande `cd ..` vous ramènera dans votre répertoire de travail.

Il existe toutefois un raccourci pour vous rendre dans votre répertoire de travail, depuis n'importe quel endroit dans le système de fichiers : `cd` tout court. `cd` sans arguments est équivalent à `cd` suivi du chemin d'accès absolu de votre répertoire de travail.

De plus, si vous donnez un tiret au lieu d'un nom de répertoire, vous reviendrez au répertoire où vous étiez juste avant.

```
$ cd
$ pwd
/u/semunix
$ cd PROJET
$ pwd
/u/semunix/PROJET
$ cd ..
$ pwd
/u/semunix
$ cd -
$ pwd
/u/semunix/PROJET
```

3.5.4 ls

Maintenant que nous savons nous déplacer d'un répertoire à l'autre, il serait utile de pouvoir inspecter le contenu de ces répertoires ! C'est exactement le rôle de la commande `ls`

Windows

Pour les habitués de DOS, `ls` est l'équivalent du *« dir »*

.

Déplacez-vous dans votre répertoire de travail et entrez la commande `ls`. Vous verrez une liste de noms s'afficher : voila le contenu de votre répertoire de travail.

```
$ ls
Desktop     ns_imap
PROJET      nsmail
```

Si vous voulez obtenir plus d'information sur ces fichiers, ajoutez l'option **-l** (*« long »*) à la commande :

```
$ ls -l
total 2531
```

```
drwxr-xr-x    4 semunix   etud          1024 Jul 23 00:32 Desktop
drwx------    3 semunix   etud          1024 Jul 23 00:33 ns_imap
drwx------    2 semunix   etud          1024 Jul 23 00:34 nsmail
```

Remarquez que les fichiers sont en ordre alphabétique, avec les majuscules d'abord. Notez aussi que les répertoires et les fichiers sont mélangés, contrairement à d'autres systèmes qui montrent les répertoires d'abord.

Les informations détaillées contiennent :

drwxr-xr-x Le type et les droits d'accès du fichier.

4 Le nombre de liens sur le fichier.

semunix L'usager à qui appartient le fichier.

etud Le groupe auquel est attribué le fichier.

1024 La taille en octets du fichier

Jul 23 00 :33 La date de dernière modification du fichier.

Desktop Le nom du fichier.

Si vous ne spécifiez pas l'option **-a**, `ls` ne vous montre pas les fichiers dont le nom commence par un point. Contrairement à d'autres systèmes d'exploitation (comme Windows), il n'y a pas d'attribut *« caché »* pour un fichier. Simplement, si le nom de fichier commence par un point, c'est un fichier caché.

Windows

Contrairement à Windows, UNIX ne donne pas d'attributs *« Read-Only »*, *« Archive »*, etc. Les permissions et le point en début de nom donnent le même effet.

```
$ ls -a
.            .cshrc      Desktop     ns_imap
..           .emacs      nsmail
```

L'option **-a** révèle la présence d'autres fichiers. Notez qu'elle affiche aussi `.` et `..` qui sont, comme nous l'avons vu, des pointeurs vers le répertoire courant et vers le répertoire parent, respectivement.

Si l'on ne veut pas avoir la liste entière des fichiers, on peut donner en argument la liste de fichier que l'on souhaite afficher. On peut aussi utiliser des caractères spéciaux comme * qui remplace n'importe quelle chaîne et ? qui remplace n'importe quel caractère.

```
$ ls ns*
ns_imap:
ns_imap.conf

nsmail:
```

Si l'un des arguments est un répertoire, `ls` affichera la liste de fichiers que ce répertoire contient, sauf si on a spécifié l'option **-d**.

3.5.5 mkdir

La commande `mkdir` (*MaKe DIRectory*) sert à créer des répertoires. Comme toutes les autres commandes, elle accepte des noms de fichiers relatifs ou absolus.

Placez-vous dans votre répertoire de travail et faites la commande `mkdir devoirs`. Un `ls` vous montrera maintenant que le répertoire `devoirs` a été créé.

```
$ ls
Desktop     ns_imap      nsmail
$ mkdir devoirs
$ ls
Desktop     devoirs      ns_imap      nsmail
```

L'option **-p** (*« parents »*) permet de créer plusieurs niveaux de sous-répertoires d'un seul coup :

```
$ cd
$ mkdir -p devoirs/ift1010/tp1
$ cd devoirs/ift1010/tp1
$ pwd
/u/semunix/devoirs/ift1010/tp1
```

3.5.6 rmdir

La commande `rmdir` (*ReMove DIRectory*) sert à effacer un répertoire. Il faut que le répertoire soit complètement vide avant de pouvoir l'effacer.

```
$ cd devoirs
$ ls -a
.            ..
$ cd ..
$ rmdir devoirs
```

Notez l'usage de `ls -a` pour s'assurer que le répertoire est vide.

3.5.7 cp

La commande `cp` (*CoPy*) sert à copier un ou plusieurs fichiers. Elle prend au moins deux paramètres : la source et la destination. Son comportement varie selon que la destination est un fichier simple ou un répertoire. Si la destination est un fichier simple (ou si c'est un nom de fichier inexistant), une copie du fichier source est créée sous le nom de la destination. Si la destination est un répertoire, le fichier source est copié dans ce répertoire, avec le même nom ; la commande accepte alors plusieurs fichiers source : ces fichiers seront tous copiés dans le répertoire de destination.

Les principales options de `cp` sont :

-i Demande une confirmation avant de remplacer un fichier (*« interactif »*)

-r Permet de copier des répertoires. Sans cette option, les répertoires ne peuvent apparaître dans la liste des fichiers source (*« récursif »*)

-f Ignore les fichiers non-existants, les erreurs et ne demande aucune confirmation (*« force »*)

Voici un exemple d'utilisation :

```
$ ls
toto.txt
$ cp toto.txt tata.txt
$ ls
tata.txt   toto.txt
$ mkdir backup
$ cp toto.txt backup
$ cd backup
$ ls
toto.txt
$ cd ..
$ cp -i tata.txt toto.txt backup
cp: overwrite 'backup/toto.txt'? y
$ cp -r backup backup-2
backup    backup-2  tata.txt   toto.txt
```

3.5.8 mv

La commande `mv` (*MoVe*) sert à déplacer des fichiers. Elle est aussi utilisée pour renommer un fichier : en *« déplaçant »* le fichier d'un nom à un autre.

Son usage est identique à celui de `cp`. En fait, `mv` se comporte exactement comme `cp`, à la différence qu'une fois l'opération de copie effectuée, les fichiers originaux sont détruits.

Ainsi, pour renommer `toto.txt` en `tata.txt`, on donne la commande

```
$ mv toto.txt tata.txt
```

3.5.9 rm

La commande `rm` (*ReMove*) sert à supprimer un ou plusieurs fichiers.

Attention

Il n'y a pas de poubelle ou de *undelete* sous UNIX

Il convient d'insister sur ce point car il a fait le malheur de trop d'étudiants déjà : sous UNIX, un fichier supprimé est supprimé pour toujours : il n'y a pas de *« Corbeille »* où on peut retourner chercher les fichiers supprimés, comme plusieurs autres systèmes d'exploitation. Un `rm` est toujours

définitif. Faites donc attention à chaque fois que vous en lancez un. Il est de bon usage de s'arrêter une seconde et de relire attentivement la ligne de commande avant de la lancer.

-i Demande une confirmation avant d'effacer chaque fichier (*« interactif »*)

-r Permet d'effacer des répertoires (efface leur contenu d'abord) (*« récursif »*)

-f Ignore les fichiers non-existants, les erreurs et ne demande aucune confirmation. (*« force »*) Attention, cette option est **dangereuse** !

```
$ rm -i toto.txt
rm: remove 'a'? y
$ rm -r backup
rm: descend into directory 'backup'? y
rm: remove 'backup/toto.txt'? y
rm: remove 'backup/tata.txt'? y
rm: remove directory 'backup'? y
```

3.5.10 chmod

La commande `chmod` (*CHange MODe*) sert à modifier les permissions d'accès à un fichier. Seul le propriétaire du fichier a le pouvoir d'en changer les permissions (à l'exception du superusager, qui, comme nous l'avons vu, a tous les pouvoirs).

Il existe deux méthodes d'exprimer les changements de droits d'accès : symbolique et absolue.

Symbolique

La syntaxe de la commande en mode symbolique est la suivante :

`$ chmod [options] [qui]op<permission(s)> <fichiers>`

options

-R Effectue récursivement les changements de permissions dans les sous-répertoires.

-f Ignore les erreurs.

qui sert à dire sur quel ensemble de permissions nous agirons, soit celles du propriétaire (**u**), du groupe (**g**) ou de toutes les autres personnes (**o**). Par défaut, toutes les personnes seront sélectionnées (**ugo**). Une abréviation pour **ugo** est **a** (pour all).

u désigne le propriétaire du compte (**user**).

g désigne les personnes du groupe (**group**).

o désigne tous les autres (**others**).

op opérateur de permissions.

- Enlever des permissions.

+ Ajouter des permissions.

= Fixer les permissions.

permissions sert à spécifier les permissions traitées.

r Permission de lecture

w Permission d'écriture.

x Permission d'exécution.

<**fichiers**>* Un ou plusieurs fichiers ou répertoires sur lesquels les permissions doivent être appliqué.

Un exemple est donné un peu plus loin.

Absolue

La syntaxe de la commande en mode absolue est la suivante :

```
$ chmod mode <fichiers>*
```

mode Les modes sont définis selon le type d'usager et le type d'utilisation.

- **400** Lecture propriétaire
- **200** Écriture propriétaire
- **100** Exécution propriétaire
- **040** Lecture pour le groupe
- **020** Écriture pour le groupe
- **010** Exécution pour le groupe
- **004** Lecture pour les autres
- **002** Écriture pour les autres
- **001** Exécution pour les autres

Il faut faire la somme des droits d'accès que nous voulons donner.

<**fichiers**>* Un ou plusieurs fichiers ou répertoires sur lesquels les permissions doivent être appliqué.

Exemple

```
$ ls -l fichier.txt
-rwxr-xr-x   58 semunix  etud         5120 Jul 26 12:19 fichier.txt
$ chmod o= fichier.txt
$ ls -l fichier.txt
-rwxr-x---   58 semunix  etud         5120 Jul 26 12:19 fichier.txt
$ chmod g+rwx fichier.txt
$ ls -l fichier.txt
-rwxrwx---   58 semunix  etud         5120 Jul 26 12:19 fichier.txt
$ chmod 755 fichier.txt
$ ls -l fichier.txt
-rwxr-xr-x   58 semunix  etud         5120 Jul 26 12:19 fichier.txt
$ chmod 600 fichier.txt
$ ls -l fichier.txt
-rw-------   58 semunix  etud         5120 Jul 26 12:19 fichier.txt
```

3.5.11 echo

Affiche ses paramètres, rien de plus.

```
$ echo 1 2 3 4
1 2 3 4
```

La commande `echo` est surtout utilisée pour tester l'expansion de noms de fichiers, pour voir la valeur des variables d'environnement, ou dans des scripts. Ces sujets avancés ne sont vus que dans la section 6.2.

3.5.12 man

La plupart des logiciels ont une documentation en ligne (*on-line*). La commande **man** est ni plus ni moins votre aide-mémoire et votre mode d'emploi pour les problèmes de syntaxe et d'options disponibles des commandes unix. Une commande peut avoir une infinité d'options des plus abracadabrantes (utiles) et personne ne peut se vanter de toutes les connaître.

Rien ne sert de pleurer sur la difficulté de certaines commandes ! Vous aurez toujours la commande **man** pour vous rappeler la syntaxe d'une commande. Par exemple, pour avoir de l'aide sur la commande `ls`, tapez

```
$ man ls
```

C'est pratiquement ironique, mais même la commande **man** a des options que vous pouvez consulter en tapant

```
$ man man
```

`man` offre (entre autres) les options suivantes :

-k **apropos**
: Cherche pour le texte dans tous les fichiers d'aide et affiche le titre des rubriques qui correspondent.

-a Par défaut, **man** affiche la première rubrique d'aide. L'option **-a** demande d'afficher toutes les rubriques.

apropos

La commande **man** est très pratique lorsque nous connaissons le nom de la commande qui doit faire ce que nous voulons. Mais si nous ne connaissons pas ce nom, notre problème demeure-t-il entier ? Pas tout à fait. Parmi les fameuses options de **man**, il existe l'option **-k** qui peut intervenir.

Cette option vous permet de trouver une commande à partir d'un mot-clé. Par exemple, quelle est la commande pour imprimer ? Au lieu de faire la recherche vous-même parmi toutes les commandes de unix, laissez donc la machine faire le travail à votre place. Comme les commandes sont presque toutes en anglais, cherchons avec le mot-clé *« print »* :

```
$ man -k print
```

Vous devriez trouver votre bonheur dans la liste des commandes que le système vous retourne. `man` emploie votre *pager* par défaut pour afficher son résultat. (voir la section 4.6.1).

Sections

Le texte affiché par la commande **man** est divisé en plusieurs sections. Nous retrouvons généralement les sections suivante :

NAME	Donne le nom et une brève description des commandes décrites dans la page.
SYNOPSIS	Donne le format d'appel de la commande.
DESCRIPTION	Décrit plus en détail la commande.
OPTIONS	Donne la liste des options de la commande.
SEE ALSO	Donne des référence vers les commandes ayant un lien logique avec la commande.

3.5.13 info

La commande `info` est d'un fonctionnement très similaire à `man` : vous tapez

```
$ info ls
```

et vous obtenez un manuel sur `ls`. Appuyez sur `[q]` pour quitter `info`.

En général, les pages de documentation d'`info` sont plus détaillées, mais par contre il y en a moins.

3.5.14 Autres commandes

Avant de conclure la section des commandes, voici quelques commandes dont l'usage n'est pas détaillé et qui sont utiles, sans être essentielles.

Comme pour toute commande, n'hésitez pas à fouiller dans les pages de manuel pour connaître tous les détails.

cal Affiche un calendrier. Essayez les commandes suivantes :

```
$ cal
$ cal 2002
$ cal 1 2000
```

date Donne l'heure et la date. Permet un formatage personnalisé de la sortie, voir `man date`.

finger Pour avoir de l'information sur l'usager joebleau, faites `finger joebleau`.

rwho Permet d'avoir la liste des personnes présentement branchées au réseau.

tree Affiche l'arbre des sous-répertoires du répertoire courant.

Voyez aussi la section 4.6 pour d'autres utilitaires variés, notamment des utilitaires pour filtrer et agir sur du texte.

3.6 L'environnement KDE

La plupart des systèmes d'exploitation n'ont qu'un seul environnement de travail, et une seule interface graphique. Un environnement Windows a toujours l'air d'un environnement Windows, peu importe la configuration de la machine. Sur Linux, c'est plutôt le contraire ; il existe une multitude d'environnements, chacun ayant des caractéristiques différentes, une allure particulière, et un fonctionnement différent. Nous allons maintenant voir un peu comment fonctionne l'environnement KDE, votre environnement par défaut.

L'environnement de *desktop* KDE (*K Desktop Environment*) est un des environnements sur Linux qui ressemblent le plus à Windows. C'est aussi l'un des plus répandus ; cette popularité est due au grand nombres d'options et de caractéristiques qu'il présente.

Vous avez sûrement remarqué quand vous vous branchez sur une station, une petite boîte sous votre nom et votre mot de passe vous permettant de choisir entre différentes *« sessions »* : KDE, Gnome, et autres. Vous êtes libre d'en essayer d'autres ; vous pourrez toujours revenir à KDE si elles ne vous plaisent pas. Soyez toutefois avertis que l'équipe du support technique ne donne pas d'aide pour les environnements de *desktop* autres que KDE.

3.6.1 Initialisation

La première fois que vous vous branchez sur un poste de travail, KDE vous affiche plusieurs écrans de configuration. Ces écrans n'apparaîtront que la première fois. Nous vous recommandons de choisir les options données ici ; vous pourrez toujours les modifier plus tard.

Pour chacun des écrans (le titre de chacun de trouve en haut à gauche de l'écran) sélectionnez les options suivantes :

1 - Introduction Sélectionnez *« Canada »* dans *« North America »*, puis cliquez sur *Next*

2 - I Want It My Way Cliquez sur *Next*

3 - Eyecandy-O-Meter Amenez le curseur complètement à gauche, puis cliquez sur *Next*

4 - Everybody Loves Themes Cliquez sur *Next*

5 - Configure The KDE Panel Cliquez sur *Next*

6 - Time To Refine Cliquez sur *Finish*

3.6.2 Aperçu du *desktop*

Au bas à gauche du *desktop* se trouve un bouton avec un **K** : c'est le menu principal (l'équivalent du menu *« Démarrer »* de Windows).

Deux icônes plus à droite, vous trouvez l'icône de la console, sûrement celui que vous utiliserez le plus !

L'icône représentant une maison ouvre votre répertoire de travail via l'explorateur visuel de fichiers.

Vers la droite, à gauche de l'horloge, se trouvent un bouton rouge et un bleu. Le rouge sert à se débrancher. Le bleu sert à verrouiller l'écran. Quand l'écran est verrouillé, il faut que vous entriez votre mot de passe pour pouvoir reprendre le contrôle de la machine. C'est très pratique quand vous devez vous absenter temporairement de votre poste (pour aller aux toilettes, disons).

Attention

Il est interdit de verrouiller son poste pour plus de 15 minutes

.

S'il ne reste plus de postes libres et que vous êtes **certain** qu'un poste est barré depuis plus de 15 minutes, vous êtes autorisé à débrancher la personne de force en appuyant sur Ctrl-Alt-Backspace.

3.6.3 Les *desktops* virtuels

Les systèmes de fenêtrage UNIX offrent pour la plupart une caractéristique qu'on ne retrouve pas ailleurs : les *desktops* virtuels. La plupart des systèmes d'exploitation offrent un seul *desktop*, c'est à dire un seul espace pouvant contenir des fenêtres. Sous UNIX, vous avez plusieurs *desktops* (quatre, en général) et vous pouvez naviguer de l'un à l'autre ou faire passer des fenêtres de l'un à l'autre. Cela permet d'organiser les fenêtres d'une manière très pratique en les regroupant. Vous pourriez, par exemple, placer les fenêtres de votre navigateur Web dans un desktop, votre éditeur de texte dans un autre, votre lecteur CD dans un troisième, etc.

Bien sûr, vous pouvez vous limiter à un seul *desktop*, si les multiples *desktops* vous embarrassent.

Pour changer de desktop en KDE, cliquez sur les rectangles numérotés qui apparaissent à peu près au milieu de la barre des tâches, au bas de l'écran. Vous pouvez aussi utiliser Ctrl-Tab pour alterner.

Pour envoyer une fenêtre vers un autre *desktop*, cliquez sur la barre de titre (barre bleue) avec le bouton de droite, puis allez dans *« To Desktop... »* et cliquez un des desktops dans la liste.

3.6.4 Opérations copier-coller

Sous UNIX, il est en général très simple de copier-coller : sélectionnez le texte à copier, et **sans le désélectionner** cliquez avec le bouton du milieu de la souris à l'endroit où vous voulez le coller. Si votre souris n'a que deux boutons, vous pouvez généralement cliquer les deux boutons en même temps pour émuler le bouton du milieu.

Certaines applications ont un comportement un peu différent :

Emacs Dans emacs, cliquez une fois avec le bouton de gauche pour placer le curseur, puis cliquez ailleurs avec le bouton de droite pour sélectionner (et donc copier) le texte entre les deux points. Si au lieu de cliquer avec le bouton de droite vous double-cliquez, vous coupez le texte. Utilisez ensuite le bouton du milieu pour coller.

Mozilla Pour atteindre un URL dans Mozilla vous pouvez le copier à partir de n'importe quelle application (en le sélectionnant) puis aller cliquer dans la fenêtre de Mozilla avec le bouton du milieu. Mozilla se rendra alors à cet URL.

3.6.5 Raccourcis clavier

Voici quelques raccourcis clavier utiles en KDE :

Alt-Tab Passe d'une fenêtre à l'autre

Ctrl-Tab Passe d'un *desktop* virtuel à un autre

Ctrl-Alt-Esc Ouvre `xkill` (voir section 3.7.1)

Ctrl-Alt-Backspace Force la déconnexion (voir section 3.7.1)

Alt-F4 Ferme la fenêtre active (celle au-dessus des autres)

3.7 Obtenir de l'aide

La documentation officielle du réseau informatique du DIRO se trouve sur la page web du support technique. Cette documentation est mise à jour régulièrement par l'équipe du support technique.

La documentation officielle du réseau informatique du DIRO.
`http ://support.iro.umontreal.ca/`

URL

Pour obtenir de l'aide, voici les ressources en ordre de priorité d'essai :

1. **man** et **info** (voir 3.5.12 et 3.5.13). Essayez aussi la commande avec l'option `--help`.
2. La personne à côté de vous
3. Site Web, démonstrateur et professeur du cours.
4. Le groupe semunix. `http ://www-etud.iro.umontreal.ca/~semunix/`
5. Le site de support. `http ://support.iro.umontreal.ca/faq.shtml`

3.7.1 Que faire si ca plante ?

On entend souvent dire que *« Linux plante moins souvent que Windows »*. Cette affirmation est vraie en général, et absolument en termes de comparaison des noyaux (*kernels*) des deux systèmes. Toutefois, la qualité du système d'exploitation n'implique pas la qualité des logiciels qu'on y emploie : vous verrez sûrement, au cours de vos nombreuses heures d'utilisation, planter certains programmes . Que faire quand un programme gèle, ne veut plus se fermer, et que le système ne répond plus ?

N'oubliez pas que **il ne faut pas couper l'alimentation électrique de la machine** en appuyant sur le bouton de redémarrage. Un redémarrage brutal risque d'endommager le système de fichiers, et aussi de vous attirer la colère des administrateurs.

Essayez plutôt les trucs suivants :

Pour forcer un programme qui plante à quitter Ce truc est à utiliser si un programme fonctionne mal mais que le reste de l'interface fonctionne toujours (le pointeur de la souris se déplace encore).
Faites Ctrl-Alt-Esc. Votre curseur prend l'apparence d'une tête de mort. Cliquez sur le programme fou — il devrait se terminer immédiatement.
Vous pouvez obtenir la même tête de mort en tapant `xkill` (si vous êtes capable d'ouvrir un terminal, bien sur).

Si toute l'interface est gelée Si la souris et le clavier ne répondent plus, essayez Ctrl-Alt-Backspace. Cela va redémarrer le serveur X et par le fait même vous débrancher.

Si vous n'arrivez pas à vous débrancher Faites Ctrl-Alt-F4. Vous devriez alors passer en mode console. Faites alors Ctrl-Alt-Delete pour redémarrer la machine.

Si rien de tout cela ne fonctionne Après avoir essayé toutes ces options, et si personne ne peut vous conseiller, vous n'avez d'autre choix que de redémarrer la machine en appuyant sur le bouton *« reset »*.

Chapitre 4

Logiciels disponibles

Cette section donne un aperçu des logiciels les plus pratiques et les plus fréquemment utilisés sous Linux et au DIRO.

Nous ne pouvons donner une documentation exhaustive de chaque programme dans ce manuel ; n'oubliez jamais que vous pouvez obtenir de l'aide au sujet de la quasi-totalité des programmes avec la commande `man` (section 3.5.12).

4.1 Internet et WWW

L'Internet et en particulier le Web s'avèreront sans aucun doute des ressources précieuses lors de vos études. Voici plusieurs programmes qui vous seront utiles pour en profiter.

4.1.1 Navigateurs Web

Il existe une panoplie de navigateurs web pour la plateforme Linux. Parmi les plus utilisés, on retrouve Firefox, Mozilla, et Lynx. Les deux premiers ont des interfaces similaires à celles qu'on retrouve sur les autres plateformes. En général, quelqu'un qui est habitué à un fureteur en particulier peut passer à un autre sans trop de problèmes : l'usage est le même. Chaque navigateur a toutefois ses particularités : ce sont ces particularités que nous verrons ici.

Lynx mérite une discussion séparée car, contrairement aux autres, il fonctionne en mode texte seulement, et par conséquent son interface ne ressemble à rien de populaire.

Bases communes

Le fonctionnement de base d'un navigateur est assez simple et, surtout, connu de presque tous. Nous ne nous y attarderons pas. Ceux qui sont peu familiers avec la navigation du Web peuvent aller lire les tutoriels suggérés plus bas.

Le navigateur web
`http ://www.commentcamarche.net/www/navigateur.php3`

URL

Pour un exemple expliquant comment imprimer une page web avec Firefox, voir la section 4.5.1.

Mozilla

Mozilla est un des projets phare de la communauté *« Open Source »*. En 1998, quand Netscape rendit public le code source de Navigator, un groupe de volontaires se forma (principalement d'employés de Netscape) et entreprit de créer un navigateur *Open-Source*.

Mozilla se targue d'être le navigateur le plus respectueux des standards, un critère de plus en plus populaire à cause des grandes divergences dans le monde du web qui rendent la compatibilité entre sites et fureteurs difficile.

Mozilla offre les avantages suivants :

- Permet de bloquer les *pop-ups*, ces petites fenêtres contenant de la pub qui apparaissent souvent ;
- La sécurité - les concepteurs le vantent comme étant le fureteur le plus sécuritaire ;
- Permet de visionner plusieurs pages dans la même fenêtre à l'aide d'onglets
- Permet de zoomer sur le texte des pages web.

par contre, son désavantage majeur est son interface plutôt lourde (longue à charger).

Pour lancer Mozilla, tapez la commande

```
$ mozilla &
```

Il se peut que parfois, quand vous ouvrez Mozilla, un dialogue comme celui de la figure 4.1 vous apparaisse. Ce dialogue indique que soit vous avez oublié de vous débrancher à une autre machine, soit Mozilla a planté à la dernière utilisation.

Si le dialogue vous apparaît à répétition, vous devez donner la commande suivante :

```
$ rm ~/.mozilla/lock
```

Parmi les avantages de Mozilla, on retrouve :

- Respect des standards pour un meilleur rendu des sites ;
- Permet de bloquer les *« pop-ups »* ;
- Permet de visionner plusieurs pages dans la même fenêtre (ce qu'on appelle une *Multiple Document Interface*) ;
- Permet de zoomer sur les pages web (texte seulement - les images restent de la même taille) ;
- Permet de faire des recherches à partir de la barre d'adresse.

Pour plus de détails, voyez le site web officiel :

URL

Site Web officiel du projet Mozilla
`http ://www.mozilla.org/`

Si vous êtes intéressé, jetez aussi un coup d'œil aux nombreux projets s'intégrant à Mozilla :

URL

Mozilla Projects
`http ://www.mozdev.org/projects.html`

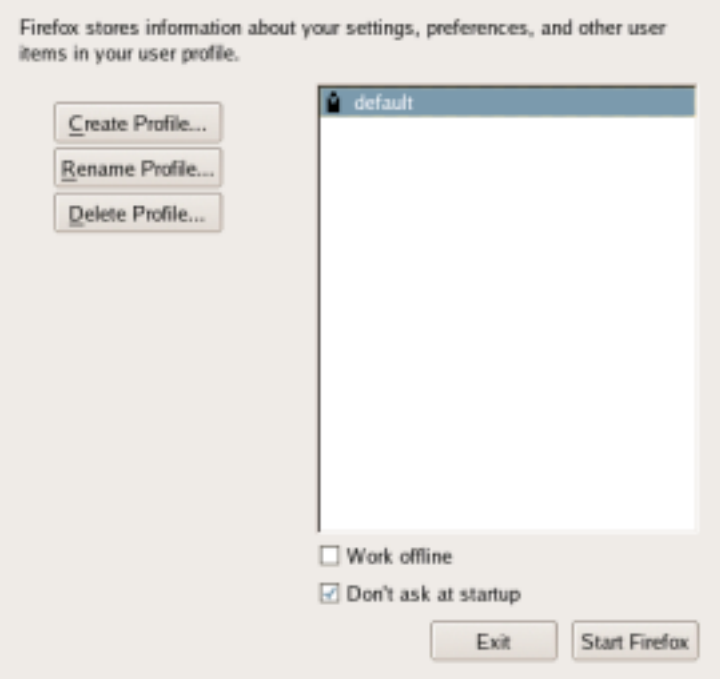


FIG. 4.1 – Ce dialogue indique que soit vous avez oublié de vous débrancher à une autre machine, soit Mozilla a planté à la dernière utilisation.

Firefox

Firefox est un projet qui dérive de Mozilla. En fait à ses débuts, il n'était qu'une version allégée de Mozilla qui ne contenait que le fureteur. Au cours de son évolution, Firefox a prit un peu de distance par rapport à son projet parent mais il offre relativement les mêmes avantages et son interface est restée toujours très similaire. En plus des avantages décrits dans la section de Mozilla, Firefox offre la possibilité de lire les nouvelles par RSS et est plus rapide à démarrer.

Pour lancer Firefox vous n'avez qu'á tapper la commande suivante :

```
$ firefox &
```

Lynx

Lynx est un navigateur très différent : il n'a pas d'interface graphique. Vous n'utiliserez donc pas la souris pour naviguer mais uniquement le clavier ; vous ne pourrez pas voir les images. Le premier contact avec Lynx est toujours rigolo car il donne un regard très différent sur les sites auxquels on est habitué (voir la figure 4.2).

Les avantages d'un tel navigateur sont entre autres la vitesse accrue puisqu'on ne télécharge pas les images, les scripts, les feuilles de style etc. Lynx permet aussi aux usagers *« techniques »*

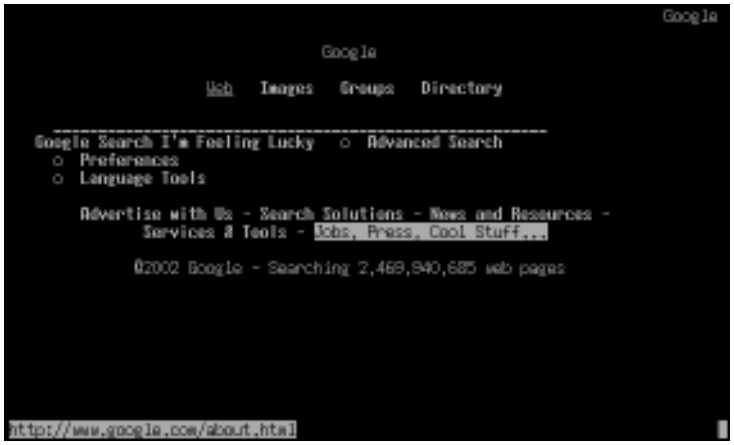

Google
Google
Web Images Groups Directory
Google Search I'm Feeling Lucky ○ Advanced Search
○ Preferences
○ Language Tools
Advertise with Us - Search Solutions - News and Resources -
Services & Tools - Jobs, Press, Cool Stuff...
©2002 Google - Searching 2,469,940,685 web pages
http://www.google.com/about.html

FIG. 4.2 – Google vu par Lynx

d'inspecter la constitution d'une page sans être embêtés par les scripts et *plugins*.

Toutefois puisqu'il présente les sites d'une façon très différente de celle habituelle, la navigation de plusieurs sites sera difficile, voire même impossible : certains sites n'utilisent que les images ou les *plugins* (Flash, par exemple) pour la navigation.

Comme l'usage de Lynx est assez unique, il convient d'en expliquer brièvement l'usage. Pour lancer Lynx, tapez la commande

```
$ lynx
```

Tapez la touche `[g]` (pour *« Go »*) puis l'adresse d'un site. Nous recommandons de commencer avec un site simple, compatible avec Lynx ; essayez `http ://www.google.ca`.

Pour visionner la page précédente/suivante, appuyez sur `[pageUp]`/`[pageDown]`. Utilisez les flèches haut/bas pour déplacer le *« focus »* d'hyperlien. Pour suivre un lien, amenez-y le focus et appuyez sur la flèche droite. Utilisez la flèche gauche pour revenir en arrière. Faites `[q]` pour quitter.

Vous pouvez maintenant découvrir Lynx - faites `[o]` pour changer les options, et `[h]` pour lire l'aide (très complète).

elinks

Un autre fureteur en mode texte ayant une interface relativement semblable à celle de lynx est ELinks. Ce dernier est plus complexe que Lynx puisque vous pouvez redéfinir les touches pour personnaliser l'utilisation. De plus, ELinks supporte les *« frames »* ce que Lynx ne fait pas. Il a même un support de base des styles graphiques.

Pour lancer ELinks tapez la commande :

```
$ elinks
```

Les touches de navigation sont sensiblement les mêmes que dans Lynx en plus de [tab] pour changer de *« frame »*. L'aide ne se trouve pas au même endroit non plus. Pour lire l'aide de ELinks, tapez [Escape] ou encore [F9] qui affichera le menu en haut de l'écran. De là vous n'avez qu'à vous diriger sur *« Help »* et à appuyer la flèche en bas, puis choisir la section d'aide que vous voulez lire. Par exemple *« Keys »* vous renseignera sur les touches et leurs utilitées.

Pour en savoir plus sur ELinks, vous pouvez consulter la page officielle du projet :

ELinks homepage
`http ://elinks.or.cz/`

URL

4.1.2 Courrier électronique et News

Plusieurs programmes de gestion de courrier électronique sont disponibles sous Linux. Certains s'apparentent aux populaires Outlook et Lotus Notes ; d'autres ont un *look* plus particulier.

Il est maintenant possible de lire son courriel dans la fenêtre d'un navigateur Web (Webmail) : `http ://webmail.iro.umontreal.ca/`

Nous allons concentrer notre étude sur les deux lecteurs de courrier suivants : mutt et Thunderbird.

mutt

mutt fonctionne uniquement en mode texte. *« Pourquoi se compliquer la vie avec une interface aussi peu invitante ? »* me demanderez-vous, et à cette question, je répondrai que ce programme s'avère très pratique lorsqu'on ne peut que se connecter au département qu'à l'aide de ssh (par exemple, chez un ou une amie). Il est également très léger sur le processeur et part presqu'instantanément. De plus, il est possible de le configurer pour qu'il adopte des comportement, des allures et mêmes les raccourcis clavier que vous désirez.
Pour démarer mutt, il vous suffit de tapper dans un terminal la commande :

```
$ mutt
```

Avec la configuration par défaut de cette application, la ligne du haut indique toujours les commandes les plus fréquentes, précédées de la lettre du clavier qui correspond à la commande. La dernière ligne du bas est normalement vide, sauf pour afficher les messages d'erreur. L'avant-dernière ligne, elle, donne des informations sur l'emplacement des messages, le nombre total de messages, le critère de rangement et la position dans la liste des messages représenté en pourcentage.
Au milieu de ces lignes se trouvent vos e-mails. De gauche à droite, vous pouvez y retrouver les informations suivantes sur chaque ligne : le numéro du message (utile pour sauter rapidement à un autre message), le statut du message (par exemple N pour nouveau, O pour un message qui n'a jamais été ouvert), la date de réception, le nom de l'expéditeur, la grosseur du fichier entre parenthèses et finalement, le titre du message.

Une ligne dans la liste des messages est surlignée en bleu. C'est en quelque sorte le curseur dans le programme. Vous pouvez le déplacer grâce aux flèches vers le haut et vers le bas. Pour lire un message, positionnez votre curseur sur celui-ci et appuyez sur [Entrer]. La zone du milieu se

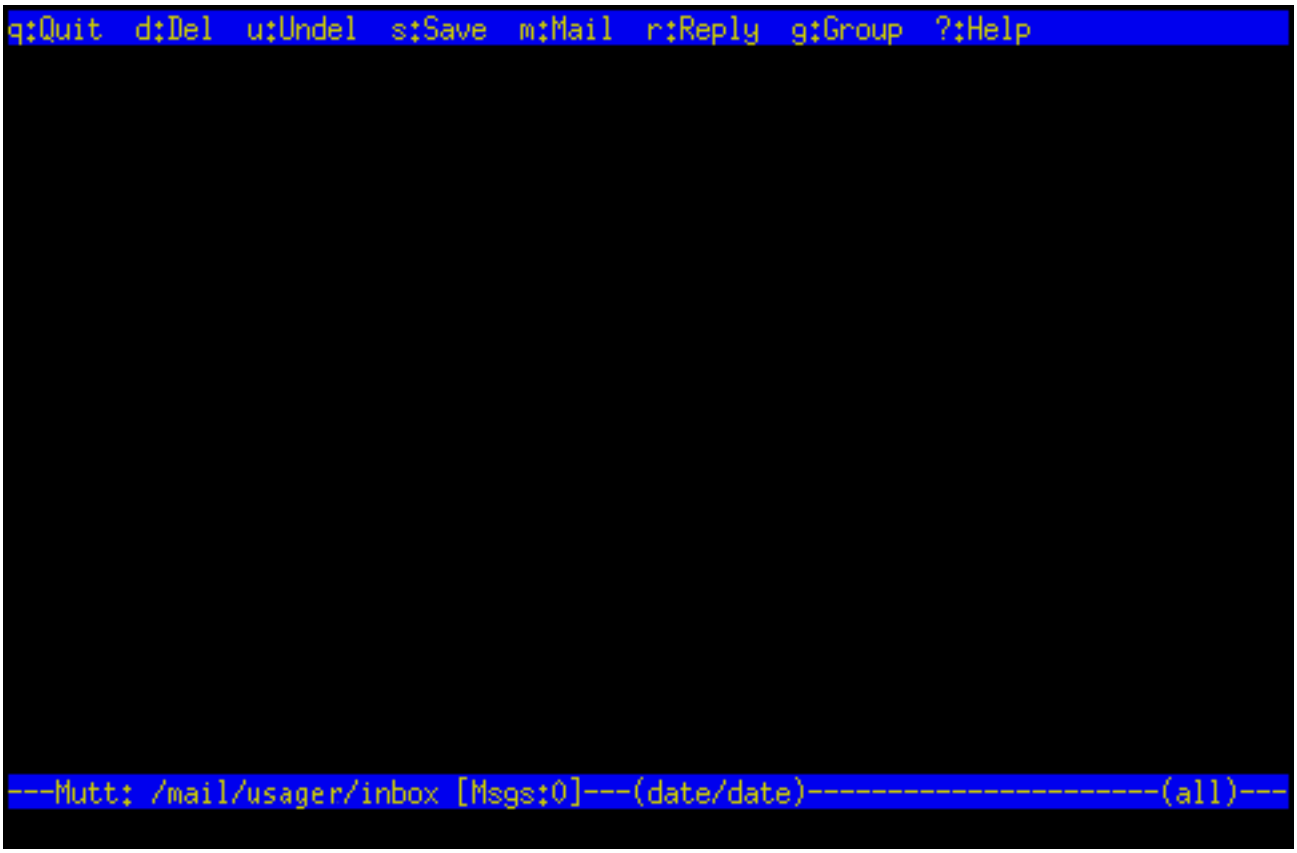


FIG. 4.3 – Écran de départ de mutt

transforme alors en l'affichage du message. Utilisez `[Page Up]` et `[Page Down]` pour voir une page plus haute et une page plus basse du message, respectivement. Pour revenir à la liste des messages, appuyez sur `[q]`. Pour effacer un message, tant dans la liste des messages que lors de l'affichage du e-mail, appuyez sur `[d]` (bien faire attention que cette lettre est en minuscule et qu'elle diffère de la commande du D majuscule). Pour composer un message, dans la liste des mails, appuyez sur `[m]` et entrez l'adresse du destinataire, puis le sujet et vous serez amenés à tapper votre message dans une interface comme celle de **Vi**.

Thunderbird

Mozilla Thunderbird est un projet qui a démarré environs en même temps que 4.1.1. Il faisait partie de l'objectif de séparer les différentes sections de Mozilla. Son interface est trés similaire à celle d'Outlook pour ceux qui y sont déjà familier. Thunderbird offre un moyen simple de filtrer les messages selons certains critéres ainsi qu'une reconnaissance de pourriels. Il offre aussi un moyen de lire les nouvelles via RSS.

Pour en savoir plus sur Thunderbird, vous pouvez vous rendre sur le site du projet :

Mozilla Thunderbird, page de support
`http ://www.mozilla.org/support/thunderbird/`

URL

Configuration du courriel au DIRO
`http ://support.iro.umontreal.ca/courriel.shtml`

URL

Autres gestionnaires de courrier

Parmi les autres gestionnaires qui s'offrent à vous, nous vous suggérons :

mutt Un autre lecteur de courriel en mode texte. Son interface est relativement simple. Une liste de commandes utiles est toujours affichée au bas de l'écran et pour plus de détails, vous pouvez taper `[?]`.

kmail Démarre avec la commande `kmail`. De style Outlook.

Mozilla Messenger Démarrez le module de courriel de Mozilla : `mozilla -mail &`

4.1.3 Transfert de fichiers

Il est souvent pratique de copier un fichier d'une machine à une autre. Par exemple, si vous écrivez un devoir sur votre ordinateur à la maison, vous devez ensuite l'amener dans votre compte pour pouvoir le remettre. Il existe plusieurs façons de transférer les fichiers. La plus populaire et plus pratique, bien que non encryptée et donc peu sécuritaire, est le FTP, dont nous verrons l'usage. Nous verrons aussi SCP, un moyen sécuritaire mais moins pratique que FTP.

ftp

FTP est un acronyme pour *« File Transfer Protocol »*. Ce protocole est standard et bien établi, et vous avez donc accès à une panoplie de programmes sur des nombreuses plateformes différentes.

L'usage du protocole est assez simple. Depuis votre poste de travail, vous utilisez un programme de *« client FTP »* pour vous connecter à un serveur. Au DIRO, c'est votre serveur attitré (voir la section 5.3.2) qui sert de serveur FTP. En début de connexion, vous devez entrer votre nom d'usager et votre mot de passe. Au DIRO, ce sont les mêmes que quand vous vous connectez à un poste.

Il existe un compte particulier appelé **anonymous**, utilisé par les serveurs ouverts au grand public. Il n'est pas rare que des sites n'acceptent que les connexions anonymes : plusieurs serveurs FTP ne servent qu'à distribuer des fichiers au grand public, et donc permettent à quiconque de se connecter. En revanche, certains sites n'acceptent les connexions que d'usagers connus du système. En général, les usagers anonymes ne peuvent pas modifier le contenu du serveur. Quand vous vous connectez anonymement, vous devez donner votre adresse email comme mot de passe.

Une fois connecté, vous pouvez vous promener dans le système de fichiers du serveur comme si ce système de fichiers était local. Vous pouvez envoyer (*upload*) et recevoir (*download*) des fichiers.

Attention

Les connexions FTP ne sont généralement pas encryptées ; un tiers pourrait donc intercepter la communication sans que vous ne vous en rendiez compte et lire votre mot de passe.

On peut séparer les programmes clients FTP en deux catégories : ceux de la ligne de commande et ceux à interface graphique, qui fournissent une interface du style de l'Explorateur Windows ou du Finder de MacOS.

Nous verrons d'abord le client le plus simple, le programme `ftp`. Ce programme est disponible sur Windows et sur Linux, bien que les deux versions comportent de subtiles mais importantes différences.

Pour débuter une session lancez la commande

```
$ ftp nom-de-serveur
```

Par exemple : `ftp frontal.iro.umontreal.ca` ou `ftp ftp.nvidia.com`

Windows

Pour démarrer le programme `ftp` sous Windows, cliquez *« Run »* dans le menu *« Démarrer »* et tapez votre ligne de commande, `ftp frontal.iro.umontreal.ca` par exemple.

Si tout se passe bien, la connexion s'établit et le programme vous demande votre nom d'usager. Sinon, il se peut que le serveur soit temporairement hors-ligne ou qu'il soit trop chargé et n'accepte plus de connexions. Dans ces deux cas, ré-essayez plus tard.

Entrez votre nom d'usager et votre mot de passe. Vous êtes maintenant libre de vous déplacer en utilisant la commande `cd`, et voir le contenu des répertoires avec la commande `ls`.

Windows

Sur Windows, vous devez taper `bin` avant de commencer les transferts de fichiers, à moins que vous ne comptiez transférer que des fichiers texte.

Si vous voulez recevoir un fichier, tapez `get nom-de-fichier`, tandis que `put nom-de-fichier` vous permet d'envoyer un fichier local sur le serveur. Lorsque vous désirez terminer la session, tapez **bye**.

Plusieurs commandes sont disponibles dans l'interface texte, la fonction **help** vous permettra d'en savoir plus. Voici un aperçu des commandes plus fréquemment utilisées :

help	vous donne la liste des commandes disponibles. Faites **help** nom-de-commande afin d'obtenir une courte description d'une commande en particulier
open nom-de-serveur <port>	ouvre une connexion à un serveur ftp. Le port par défaut (celui utilisé si vous n'en spécifiez pas un) est 21.
close	Termine la session ftp avec le serveur

bye	Termine la session ftp avec le serveur et ferme le programme
cd nom-de-répertoire	Change le répertoire courant sur le site
lcd	Change le répertoire courant local
ls	Liste le contenu du répertoire courant du site
get nom-de-fichier <nom-de-fichier-local>	Copie le fichier du serveur dans le répertoire courant local. Vous pouvez spécifier facultativement un nouveau nom.
mget nom-de-fichiers	Copie plusieurs fichiers à la fois. Vous pouvez utiliser le caractère * un peu comme dans le shell
put nom-de-fichier <nom-de-fichier-distant>	Copie le fichier local vers le site ftp. Vous pouvez spécifier facultativement un nouveau nom.
mput nom-de-fichiers	Copie plusieurs fichiers à la fois. Vous pouvez utiliser le caractère *
ascii	Change le type de transfert pour le mode texte. Les différents systèmes d'exploitation utilisant des formats différents de fichiers texte (notamment pour les caractères de fin de ligne), ce mode de transfert fera la conversion entre les différents formats si nécessaire. À taper avant de commencer à transférer des fichiers texte.
binary	Place le type de transfert en mode binaire. De cette façon, aucune conversion n'est effectuée. À taper avant de transférer des fichiers qui ne sont pas du texte.

Si vous aimez faire les transferts FTP depuis la ligne de commande, vous apprécierez sûrement le programme `ncftp` qui offre toutes sortes d'avantages comme la complétion de noms de fichiers, les transferts en *background*, les favoris, etc. Voyez le manuel pour tous les détails.

Information

Vous préférerez peut-être les programmes à interface graphique, plus intuitifs et souvent offrant plus d'options. La plupart de ces programmes ont deux fenêtres permettant de naviguer dans les répertoires et de manipuler les fichiers. Généralement, celle de gauche montre le système de fichiers local et celle de droite le système de fichiers distant. Nous n'expliquerons pas l'usage de ces clients ici, mais vous suggérons d'essayer `gftp`, un client graphique très pratique. Pour le démarrer faites simplement

```
$ gftp &
```

scp

scp constitue un moyen simple et sécuritaire (encrypté) de transférer des fichiers d'une machine à une autre. Son usage est à peu près identique à celui de `cp`, mais un nom de machine et d'usager

peut être ajouté aux noms de fichiers. Par exemple, depuis votre machine Linux à la maison, pour copier un fichier du département à votre machine, vous feriez :

```
$ scp semunix@frontal.iro.umontreal.ca :/u/semunix/monfichier .
```

où *« semunix »* est votre nom d'usager, *« frontal.iro.umontreal.ca »* est le nom de votre serveur et *« /u/semunix/monfichier »* est le nom absolu du fichier à transférer.

Il est également possible de transférer un fichier d'un serveur distant à un autre. Il suffit d'inclure un nom d'usager et de machine pour la source ainsi que pour la destination.

Comme ce mode de transfert est encrypté, nous vous conseillons d'utiliser ce dernier pour vous protéger, surtout si vous êtes dans un lieu publique, comme par exemple sur votre portable dans un café.

Plus de détails dans la page de manuel de `scp` ainsi que dans la section 6.3.4.

4.1.4 Connexion à distance

Il existe plusieurs manières d'ouvrir un terminal sur une machine à distance. Les programmes qui permettent de le faire sont très simples à décrire puisqu'ils sont très transparents : une fois la connexion effectuée, le terminal devient simplement *« à distance »*. Il fonctionne de la même façon qu'un terminal local, mais permet de manipuler des fichiers et exécuter des programmes à distance.

Voyez la section 6.3 pour plus de détails pratiques concernant la connexion depuis votre domicile.

ssh

SSH est le moyen par excellence pour une connexion à distance. Son usage est assez simple :

```
$ ssh login@machine
```

Si vous omettez le paramètre <login>, votre nom d'utilisateur local sera utilisé (ne tapez pas le *« @ »* si vous omettez le nom d'utilisateur).

Si c'est la première fois que vous vous connectez à la machine, le message suivant vous apparaîtra :

```
$ ssh frontal.iro.umontreal.ca
The authenticity of host 'frontal.iro.umontreal.ca (132.204.24.77)'
can't be established.

RSA1 key fingerprint is 30:10:0f:18:4e:fa:6c:07:f8:13:75:be:f9:cc:fb:2d.

Are you sure you want to continue connecting (yes/no)? yes
```

Vous devrez taper `yes` en toutes lettres pour effectuer la connexion.

4.1.5 Autres outils

wget et curl

`wget` et `curl` sont deux outils pratiques pour télécharger des fichiers du Web sans utiliser un navigateur. Cela peut être pratique dans plusieurs contextes, principalement dans des scripts ou quand le navigateur traite le fichier d'une manière peu pratique. Par exemple, quand vous ouvrez un vidéo via Mozilla, le *plugin* prend immédiatement le contrôle, et ne vous donne pas d'option pour sauvegarder le vidéo sur votre ordinateur. Une fois que vous avez l'URL du fichier, le télécharger est un jeu d'enfant avec `wget` ou `curl`.

Pour sauvegarder une page Web ou un fichier accessible via le Web, il suffit d'appeler `wget` avec l'URL du fichier. Par exemple, pour sauvegarder le logo de Google :

```
$ wget http ://www.google.com/images/logo.gif
```

`wget` peut aussi être utilisé pour télécharger plusieurs pages Web d'un coup : le programme va chercher une première page (que vous lui spécifiez), l'analyse pour en extraire les liens vers d'autres pages, puis va chercher ces autres pages, et recommence le processus. On appelle ce genre de programmes des *web spiders*. Par exemple, la commande

```
$ wget -r -l 1 http ://www.iro.umontreal.ca/
```

va télécharger toutes les pages qui sont directement liées à la page d'accueil du département. Utilisez cette commande avec prudence ! Vous pourriez télécharger sans vous en apercevoir d'immenses quantités d'information.

Voyez la page manuel de `wget` pour plus d'informations.

`curl` donne à peu près les mêmes fonctionnalités. Il ne supporte pas la récursion (téléchargement de plusieurs fichiers automatiquement) mais offre plus d'options concernant le transfert, notamment plusieurs options de configuration HTTP qui le rendent très utile dans les scripts. Voyez le manuel pour plus d'informations.

ping

La commande `ping` sert à tester le réseau ; elle tente d'échanger des petits paquets d'information avec une autre machine et indique si l'échange a fonctionné et combien de temps il a pris. Par exemple, pour savoir si `www.yahoo.com` fonctionne, essayez ceci

```
$ ping -c 5 www.yahoo.com
PING www.yahoo.akadns.net (64.58.76.224) from 132.204.26.158 : 56(84) bytes of data.
64 bytes from w3.dcx.yahoo.com (64.58.76.224): icmp_seq=0 ttl=46 time=25.364 msec
64 bytes from w3.dcx.yahoo.com (64.58.76.224): icmp_seq=1 ttl=46 time=27.835 msec
64 bytes from w3.dcx.yahoo.com (64.58.76.224): icmp_seq=2 ttl=46 time=25.800 msec
64 bytes from w3.dcx.yahoo.com (64.58.76.224): icmp_seq=3 ttl=46 time=29.280 msec
64 bytes from w3.dcx.yahoo.com (64.58.76.224): icmp_seq=4 ttl=46 time=25.650 msec

--- www.yahoo.akadns.net ping statistics ---
```

```
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/mdev = 25.364/26.785/29.280/1.537 ms
```

On peut voir que `www.yahoo.com` (qui a l'adresse IP `64.58.76.224`) est accessible, et que le transfert aller-retour prend en moyenne 26 millisecondes. L'option `-c 5` spécifie d'essayer d'envoyer 5 paquets seulement. Si elle n'est pas spécifiée, la commande enverra des paquets indéfiniment ; il faut alors l'arrêter avec Ctrl-C.

4.2 Éditeurs de texte

4.2.1 Emacs

Cette section a pour but de présenter un aperçu de l'éditeur Emacs, l'éditeur de texte le plus populaire au DIRO. Emacs a été conçu par des programmeurs pour des programmeurs ; il contient de nombreuses fonctions qui facilitent grandement l'édition de code source. Il sera donc un atout majeur pour vos devoirs de programmation !

À la fin de cette section, vous devriez être capable d'éditer efficacement des fichiers.

Introduction

Qu'est-ce que Emacs ? Le *Emacs HowTo* (voir le lien à la fin de cette section) donne les définitions suivantes pour Emacs :

- un éditeur de texte ;
- un gestionnaire de courrier électronique ;
- une religion ;
- un environnement de développement intégré ;
- tout ce que vous voulez.

Emacs est fondamentalement très flexible : vous pouvez transformer complètement la manière dont Emacs fonctionne (nous verrons comment). En réalité, Emacs est un interprète du langage *« Lisp »*. À la base du programme se trouve un interprète Lisp — la majeure partie du programme est écrite en Lisp, un langage assez simple. Quand vous aurez acquis assez de connaissances, vous pourrez modifier ces programmes afin de personnaliser Emacs à votre goût. C'est là que réside toute la puissance de Emacs : ce n'est pas vous qui êtes lié aux contraintes du programme, c'est le programme qui suit les contraintes que vous lui imposez.

Avant d'en apprécier toutes les subtilités, il est important d'en apprendre les rudiments, comme tout autre éditeur. Ces quelques pages devraient vous permettre d'utiliser Emacs comme n'importe quel autre éditeur simple. Emacs est un éditeur qui fonctionne aussi bien en mode graphique qu'en mode texte - ceci est particulièrement utile lorsque nous accédons au réseau via un modem à basse vitesse. Ce petit texte vous introduit aux deux modes. Même si plusieurs opérations se font avec la souris et les choix dans les menus, il est pratique et rapide de se familiariser avec les commandes au clavier.

Appel de Emacs

```
$ emacs [fichier]*
```

Raccourci et notation

Les commandes Emacs font souvent intervenir la touche CONTROL (`[ctrl]`) ou la touche META (`[alt]` ou `[esc]`). Plutôt que d'utiliser ces noms longs à chaque fois, nous utiliserons les abréviations suivantes :

`C-`*« chr »* signifie maintenir la touche CONTROL appuyée en même temps que vous tapez le caractère *« chr »*. Ainsi, `C-f` se dirait : maintenez la touche CONTROL et tapez `[f]`.

`M-`*« chr »* M-*« chr »* veut dire soit taper la touche META (souvent `[alt]`) en même temps que *« chr »*, soit taper `[esc]`, relâcher `[esc]`, puis la touche *« chr »*.

Voici des exemples de séquence de touches écrits sous la notation employée par emacs.

`C-x` Appuyez simultanément sur `[ctrl]` et `[x]`.

`C-x y` Appuyez simultanément sur `[ctrl]` et `[x]`, relâchez, et ensuite `[y]`.

`C-x C-y` Appuyez simultanément sur `[ctrl]` et `[x]`, maintenez `[ctrl]` et lâchez `[x]`, et ensuite appuyez `[y]`.

`M-x` Appuyez simultanément sur `[alt]` et `[x]`.

`M-x y` Appuyez simultanément sur `[alt]` et `[x]`, relâchez, et ensuite `[y]`.

L'interface

fichier ***file***
Un fichier est un fichier actuellement sur le disque. Vous n'éditez jamais un fichier mais vous éditez un tampon contenant une copie du fichier.

tampon ***buffer***
Un tampon est une structure de données interne que vous éditez. La plupart des tampons, mais pas tous, sont associés à un fichier. Pour éditer un fichier, vous le chargez dans un tampon ; après l'édition, vous sauvegardez ce tampon dans le fichier. Éditer ou détruire un tampon n'affecte pas le fichier tant qu'on ne sauvegarde pas le tampon. Cette terminologie peut sembler étrange mais le principe n'est pas différent des autres éditeur courants.

fenêtre ***window***
Une fenêtre est une vue sur un tampon. Il est possible d'avoir plusieurs vues sur le même tampon.

fenêtre graphique ***frame***
Une fenêtre graphique est comme une fenêtre mais est considéré comme une entité différente sous X Window. Nous ne discuterons pas des fenêtres graphiques.

minibuffer Tampon d'une seule ligne qui ne sert pas à visiter des fichiers mais à entrer des commandes et à communiquer avec Emacs

Buffers Files Tools Edit Search TeX Help
globale d'un syst\'eme \unix\ et \'a vous permettre de vous y d\'
d'y travailler. Comme vous le verrez bient\^ot, \unix\ est un sys
complexe qui ne peut \^etre r\'esum\'e en quelques pages. L'info
contenue dans ce chapitre devrait toutefois \^etre suffisante pou
permettre de vous lancer dans l'apprentissage de cet environment
\section{Philosophie}
\label{basesunix:philo}
Lorsque nous travaillons avec \unix, il faut toujours garder en t
certaines notions qui ont inspiré ses créateurs. Ces notions se r
tant dans chaque programme \unix\ que dans leur agencement.
\begin{itemize}
\item
Chaque programme ne fait qu'une seule fonction simple et il l
-:--- bases_unix.tex Fri Aug 30 09:33 0.05 Mail (TeX Hscr)--L1
\item[\emacskey{M-\suquote{chr}}]
M-\suquote{chr} veut dire soit taper la touche META (souvent
en m\^eme temps que \suquote{chr}, soit taper \sukey{esc}, re
\sukey{esc}, puis la touche \suquote{chr}.
\end{sudescriptionarray}
Voici des exemples de s\'equence de touche \'ecrits sous la notat
par emacs.
\begin{sudescriptionarray}[\emacskey{MC-c C-cM}]
\item[\emacskey{C-x}] Appuyez simultan\'ement sur \sukey{ct
\item[\emacskey{C-x y}] Appuyez simultan\'ement sur \sukey{ct
\item[\emacskey{C-x C-y}] Appuyez simultan\'ement sur \sukey{ct
\item[\emacskey{M-x}] Appuyez simultan\'ement sur \sukey{al
\item[\emacskey{M-x y}] Appuyez simultan\'ement sur \sukey{al
\end{sudescriptionarray}
-:--- programmes.tex Fri Aug 30 09:33 0.05 Mail (TeX Hscr)--L7

FIG. 4.4 – Deux fenêtres, éditant chacune un tampon visitant un fichier

Fonctionnalités de base

Chaque action exécutée par Emacs résulte de l'appel d'une fonction Lisp. Chaque commande tapée au clavier (ne serait-ce qu'appuyer sur la flèche gauche pour faire bouger le curseur vers la gauche) appelle la fonction Lisp appropriée.

Voyons les combinaisons de touches importantes sous Emacs, ainsi que la fonction correspondante. Vous pouvez lancer toute commande en tapant `M-x` suivi du nom de la commande puis `[Entrée]`.

Commande d'Emacs

`C-x C-c` **save-buffers-kill-emacs**
Quitte Emacs.

`C-z` **iconify-or-deiconify-frame**
Suspend Emacs.

`C-g` **keyboard-quit**
Interrompt toute commande en cours. Par exemple, si vous faites `Ctrl-x`, Emacs

attend la suite de votre commande. `Ctrl-g` annule.

`C-h ?` **help-for-help**
Pour obtenir de l'aide en tout temps.

Aide sous Emacs

Pour obtenir de l'aide dans emacs faite `C-h` *« chr »* où *« chr »* est un caractère ci-dessous.

`C-h ?` **help-for-help**
Pour obtenir de l'aide sur l'aide.

`C-h a` **command-apropos**
Permet d'effectuer une recherche dans la liste des commandes d'Emacs.

`C-h b` **describe-bindings**
Liste les fonctions associées aux touches.

`C-h f` **describe-function**
Décrit le fonctionnement d'une fonction d'Emacs.

`C-h k` **describe-key**
Tapez une suite de touches et Emacs décrira la commande associée à la séquence de touche.

`C-h m` **describe-mode**
Décrit le mode courant.

Le curseur

Emacs comprend de nombreux raccourcis pour naviguer plus rapidement dans le texte.

`C-a` **beginning-of-line**
Déplace le curseur au début de la ligne.

`C-e` **end-of-line**
Déplace le curseur à la fin de la ligne.

`M-f` ou `M-[→]` **forward-word**
Avance d'un mot

`M-b` ou `M-[←]` **backward-word**
Recule d'un mot.

Édition

`C-x u` ou `C-_` **undo**
Permet de défaire l'effet de la dernière commande.

`C-d` **delete-char**
Efface le caractère vis-à-vis le curseur.

`[backspace]` **DEL**
Efface le caractère qui précède le curseur.

`C-k` **kill-line**
Détruit les caractères suivant le curseur jusqu'à la fin de la ligne courante. Si le curseur est en fin de ligne, la ligne suivante vient se fusionner avec la ligne courante.

`C-w` **kill-region**
Détruit les caractères sélectionnés (*« coupe »*).

Opérations sur les fichiers

`C-x C-f` **find-file**
Pour ouvrir un fichier. Entrez le nom du fichier dans la zone du minibuffer.

`C-x C-s` **save-buffer**
Pour sauvegarder les modifications apportées à un fichier.

`C-x C-w` **write-file**
Pour sauvegarder le tampon dans un autre fichier.

Opérations sur les tampons

`C-x b` **switch-to-buffer**
Emacs demande un nom de tampon et change le tampon courant pour le nouveau tampon. Emacs crée un nouveau tampon si vous donnez un nouveau nom mais ce tampon n'est associé à aucun fichier tant qu'il ne sera pas sauvegardé.

`C-x C-b` **list-buffers** Emacs ouvre une nouvelle *window* et donne la liste des tampons actifs avec leurs informations.

`C-x k` **kill-buffer** Emacs demande un nom de tampon et efface toutes les structures du tampon de la mémoire, le tampon devient mort. Si le tampon est modifié, Emacs demandera si vous voulez sauvegarder.

`C-x C-q` **vc-toggle-read-only** Met ou enlève l'option **lecture seulement** sur le tampon courant.

Opérations sur les fenêtres

`C-x o` **other-window**
Rend actif la prochaine fenêtre. La répétition de cette commande permet de cycler parmi les fenêtres.

`C-x 1` **delete-other-windows**
Ferme toutes les autres fenêtres excepté la fenêtre courante.

`C-x 0` **delete-window**
Ferme la fenêtre courante.

`C-x 2` **split-window-vertically**
Sépare verticalement la fenêtre en deux.

`C-x 3` **split-window-horizontally**
Sépare horizontalement la fenêtre en deux.

Fonctionnalités avancées

Mise en page

Dans les modes de langages de programmation standard Emacs peut mettre en retrait automatiquement votre code. Il suffit de taper `[TAB]` et la ligne courante est automatiquement mise en

retrait.

Vous pouvez renfoncer toute une région en la sélectionnant, puis en tapant `M-x indent-region`. Si vous recevez du code mal mis en retrait, tapez : `C-x h` ce qui sélectionne tout le contenu du tampon, puis `M-x indent-region` cela peut parfois prendre du temps, alors soyez patient.

Recherche

Pour rechercher du texte dans le tampon courant, faites `Ctrl-s` puis tapez le mot que vous voulez chercher, au fur et à mesure que vous tapez, Emacs essaie de trouver la chaîne de caractères que vous donnez. Une fois que vous l'avez trouvée, appuyez sur `Ctrl-s` à nouveau pour passer à la prochaine apparition de la chaîne. Si la chaîne n'apparaît pas Emacs dira *« Failing I-Search »*.

Pour chercher vers le haut plutôt que vers le bas, utilisez `Ctrl-r`, dont l'usage est identique à `Ctrl-s`.

Remplacer un mot par un autre

Tapez `M-%`, puis votre premier mot *« mot1 »*, `[RET]` puis le second mot *« mot2 »* et `[RET]`. Emacs placera le curseur sur la première occurrence de *« mot1 »*. Tapez `[y]` ou `[n]` pour chaque occurrence du mot ou pour confirmer toutes les occurrences tapez `[ :]`.

Syntax highlighting

Emacs peut rendre votre code plus lisible pour vous, en y ajoutant des couleurs. Il embellit la présentation. Les couleurs sont accessibles uniquement en mode graphique. Vous devez être dans un mode supportant le highlighting. Tapez `M-x font-lock-mode`.

Macro

Il arrive que vous ayez à répéter plusieurs fois une séquence de touches. Pour gagner du temps Emacs met à notre disposition un système de macro. Tapez `C-x` (, votre séquence de touches et `C-x`). Votre macro est enregistrée. Pour l'appeler tapez simplement `C-x e`.

Compiler sous Emacs

Pour compiler tapez `M-x compile`. Emacs demande la commande de compilation, entrez la commande puis `[RET]`. Un tampon donnera la sortie de la compilation et les erreurs. Pour aller à la première erreur, soit vous repérez à quelle ligne s'est produite l'erreur, et vous y allez à la main, soit vous tapez `C-x ‘`. Emacs place alors votre curseur à la ligne où s'est produite l'erreur. Corrigez, puis pour aller à la prochaine erreur retapez `C-x ‘` et ainsi de suite.

Fichier de configuration

Pour configurer Emacs, il suffit d'écrire en LISP certaines commandes dans le fichier `.emacs` qui se trouve dans votre répertoire de travail. À chaque démarrage, Emacs lira ce fichier et exécutera les fonctions qui s'y trouvent.

Un fichier `.emacs` d'exemple se trouve sur la page Web des séminaires UNIX (voir les indications à la page 3).

Pour en apprendre plus

Emacs Beginner HowTo
`http ://www.tldp.org/HOWTO/Emacs-Beginner-HOWTO.html`

URL

Manuel officiel de Emacs, très complet
`http ://www.gnu.org/manual/emacs/`

URL

4.2.2 vi

Tout comme emacs, vi est un éditeur de texte très puissant. Les différences majeures entre emacs et vi :

- vi est disponible sur toutes les machines UNIX et sur plusieurs machines Windows.
- emacs interprète un programme qui a pour fonction principale d'éditer des textes, tandis que vi est véritablement un éditeur de textes (auquel on a ajouté un système permettant de le programmer). Cela veut dire qu'ultimement, emacs est plus flexible, mais il est aussi plus lent et lourd.
- Dans emacs, la plupart des commandes sont une combinaison de touches, par exemple `[Ctrl A]` ; dans vi, les commandes sont pour la plupart des touches simples.

vimtutor

Pour trouver un tutoriel de base, vous pouvez taper la commande `vimtutor`. Le texte est en anglais. L'avantage de ce tutoriel est de se trouver dans vi, ce qui entraîne que vous pourrez tester les commandes presque en même temps que vous lisez leur description. Pour quitter vimtutor, il suffit d'appuyer sur `[Esc]` pour s'assurer d'être en mode normal puis de taper `:q !`.

Démarrage et fonctions de base

Au DIRO, nous avons une version améliorée de vi nommée VIM (Vi IMproved), que nous vous conseillons fortement d'utiliser.

Donc, pour démarrer, tapez `vim` ou, dans X Window, `gvim` pour une fenêtre graphique.

Lorsque vi démarre, il est dans le mode *« normal »*. Cela veut dire que les touches du clavier servent à entrer des commandes de manipulation du texte et à se déplacer dans le texte à l'aide des touches fléchées, ou en cas de problème de terminal, avec h (haut), j (bas), k (gauche) et l (droite). Contrairement à emacs, les modes dans vi servent à distinguer comment l'éditeur interprète les commandes (dans emacs, le mode majeur indique quel programme Emacs-LISP gère une vue d'un tampon en particulier).

Pour commencer à écrire du texte, utilisez la commande `[i]` (en minuscule). Cette commande (*insert*) passe du mode normal au mode d'entrée de texte (*insert mode*). Pour quitter le mode d'insertion et revenir au mode normal, appuyez sur `[Esc]`.

Pour quitter vim, vous devez être en mode normal et entrer une commande *« ex »* (du nom du prédécesseur de vi). Toutes les commandes ex commencent par le caractère *« : »*. Pour quitter, entrez :

```
:quit !
```

Cela va quitter sans sauvegarder votre fichier. Si vous désirez sauvegarder dans un fichier, vous devez utiliser la commande write :

```
:write monfichier
```

Une fois le nom de fichier attribué, il ne faut plus le mentionner. Ainsi, pour sauvegarder à nouveau, entrez `:write` ou, tout simplement `:w`.

Principes généraux

Les commandes de vi sont normalisées. La plupart des commandes du mode normal ont le format suivant :

[répétitions] [commande] objet

Où *« répétitions »* est un nombre qui indique combien de fois exécuter une commande sur un objet. (Certaines commandes, comme `[i]`, ne prennent pas d'objet). Lorsque seul un objet est spécifié sans commande, il y a alors un déplacement. Les commandes et les objets sont représentés par des lettres.

Par exemple, la commande d'effacement est `[d]` (*delete*). Donc, pour détruire le caractère sous le curseur, on utilise `[dl]` (rappelez-vous des touches de déplacement : `[l]` seul fait déplacer le curseur vers la droite). Effacer le caractère juste à gauche du curseur est similaire : il s'agit d'appuyer sur `[dh]`.

Truc : comme ces commandes sont employées très souvent, la commande `[x]` remplace `[dl]` et la commande `[X]` remplace `[dh]`. Ainsi, pour effacer le caractère sous le curseur, il suffit d'appuyer sur `[x]`.

Si on désire effectuer la commande plusieurs fois, on ajoute un répétiteur : ainsi, pour effacer les 3 prochains caractères, on utilise `[3x]` (ou `[3dl]`).

Principaux objets

Les touches suivantes permettent de se déplacer vers un objet lorsqu'elles sont employées seules, ou indiquent à une commande normale son domaine d'action. Elles sont mentionnées en ordre subjectif d'utilité.

h, j	ligne précédente, ligne suivante
k, l	caractère précédent, caractère suivant
w, W	(word) début du mot suivant
b, B	(back) début du mot courant (ou précédent si on s'y trouve déjà)
e, E	(end) fin du mot courant (ou suivant si on s'y trouve déjà)
0, $	début de la ligne courante, fin de la ligne courante
^	début du texte sur la ligne courante
nG	(Go) ligne n (ici, la répétition est remplacée par un numéro de ligne)

n|	colonne n (ici, la répétition est remplacée par un numéro de colonne)
fX, tX	(prochain caractère X sur la ligne courante (en excluant ou incluant ce caractère).
FX, TX	précédent caractère X sur la ligne courante (en excluant ou incluant ce caractère).
H	ligne du haut de l'écran.
M	milieu de l'écran.
L	bas de l'écran.
Ctrl-D	un demi-écran plus bas
Ctrl-F	un écran plus bas
Ctrl-U	un demi-écran plus haut
Ctrl-B	un écran plus haut

La différence entre w, b, e et W, B, E tient de la définition de ce qu'est un mot. En majuscule, on considére qu'on change de mot seulement lorsqu'on rencontre un espace (ou une tabulation). En minuscule, on considère aussi qu'on change de mot dès que l'on passe de chiffres et lettres à d'autres symboles et vice-versa. Par exemple, *« 45+3 * 3 »* est fait de 3 MOTS ou de 5 mots.

Principales commandes du mode normal

Nous avons déjà vu quelques commandes du mode normal, soit `[d]` (*delete*) et `[i]` (*insert*).

Dans les listes qui suivent, il est important de noter que les commandes en minuscule sont différentes des commandes en majuscule. Cependant, elles sont souvent apparentées. En général, les opérations effectuées par les commandes en majuscule opèrent par rapport à la ligne courante. Par exemple, `[D]` détruit jusqu'à la fin de la ligne ; `[A]` se rend à la fin de la ligne avant d'entrer en mode d'insertion, etc.

Commandes qui entrent en mode d'insertion

Les commandes qui suivent ont toutes en commun de passer du mode normal au mode d'insertion ; seule la façon de faire est différente. La commande de loin la plus importante est `[i]`, insert. Toutes les autres commandes peuvent se réduire à `[i]`, des déplacements et des effacements. Par exemple, la commande `[c]` (*change*) revient à effectuer un effacement (d) suivi d'une insertion. Pour revenir en mode normal à partir du mode d'insertion, il suffit d'appuyer sur `[Esc]` ou `[Ctrl-C]`.

a	(**append**) Entre en mode insertion après le caractère sous le curseur (cette commande ne prend pas d'objet)
A	(Append) Entre en mode d'insertion à la fin de la ligne courante (cette commande ne prend pas d'objet) [équivalent à $a]
c	(change) Détruit l'objet et entre ensuite en mode d'insertion.
cc	(change line) Détruit la ligne courante et entre ensuite en mode d'insertion.
C	(Change) Détruit jusqu'à la fin de la ligne et entre ensuite en mode d'insertion [équivalent à c$]
i	(insert) Entre en mode d'insertion avant le caractère sous le curseur
I	(Insert) Entre en mode d'insertion au début du texte sur la ligne [équivalent à ^i]

o (open) Ajoute une nouvelle ligne au-dessous de la ligne courante et entre en mode d'insertion

O (Open) Ajoute une nouvelle ligne au-dessus de la ligne courante et entre en mode d'insertion item[R] (Replace) Entre en mode d'insertion écrasante : comme le mode d'insertion, mais le texte déjà existant est écrasé au lieu d'être *« poussé »*.

s (substitute) Détruit le caractère sous le curseur et entre en mode d'insertion [équivalent à xi]

S (Substitute) Détruit toute la ligne courante et entre en mode d'insertion [équivalent à cc]

Commandes qui affectent les registres (copier-coller)

Les commandes suivantes affectent les registres de texte. Les registres sont des endroits pouvant contenir des informations de façon temporaire : ils sont détruits lorsque vous quittez vi. Les registres de texte contiennent donc du texte de façon temporaire. Un éditeur de texte normal possède un seul registre : l'endroit qui stocke le texte entre le moment où vous faites *« couper »* et *« coller »*. vi, quant à lui, en possède une multitude. Il y a 25 registres de base : le registre sans nom et les registres a à z. Le registre sans nom est utilisé quand aucun autre registre n'est spécifié.

Pour spécifier un registre, il faut préfixer la commande par `[x]`, où x est le nom du registre. Les commandes suivantes prennent donc la forme :

[x] [répétitions] commande objet
ou de façon équivalente
[répétitions] [x] commande objet
ou de façon équivalente
[x] commande [répétitions] objet

d (delete) Détruit l'objet et le place dans le registre (cut)

dd (delete line) Détruit la ligne courante et la place dans le registre (cut)

D (Delete) Détruit jusqu'à la fin de la ligne et entre ensuite en mode d'insertion [équivalent à d$]

p (paste) Colle ce qu'il y a dans le registre après le curseur ; cette commande ne prend pas d'objet.

P (Paste) Colle ce qu'il y a dans le registre avant le curseur ; cette commande ne prend pas d'objet.

y (yank) copie l'objet dans le registre

Y (Yank) copie la ligne courante dans le registre [équivalent à yy]

x (xtermine) détruit le caractère sous le curseur et le place dans le registre ; cette commande ne prend pas d'objet

X (Xtermine) détruit le caractère à gauche du curseur et le place dans le registre ; cette commande ne prend pas d'objet

Note : si vous spécifiez un nom de registre en majuscules (A-Z) lorsque vous utilisez `[d]`, `[D]`, `[y]`, `[Y]`, le texte sera ajouté à l'ancien contenu du registre au lieu de le remplacer.

Recherche et mode de ligne de commande

Les touches de recherche sont un peu particulière, car l'entrée du texte à rechercher se fait au bas de l'écran, dans la *« ligne de commande »*.

Le mode d'entrée en ligne de commande a un fonctionnement spécifique assez convivial semblable à celui du shell, du moins dans VIM (dans le vi original, les explications suivantes ne s'appliquent pas). En bref, on peut utiliser les touches `[gauche]` et `[droite]` pour se déplacer pendant même qu'on entre le texte, afin d'aller le changer, et ce tant qu'on n'a pas appuyé sur `[Enter]` pour exécuter la commande ou `[Esc]` pour retourner en mode normal sans rien exécuter.

Il est aussi possible d'accéder à l'historique des dernières recherches effectuées à l'aide des touches `[haut]` et `[bas]`. Si du texte est déjà entré sur la ligne de commande, la recherche sera limitée aux autres recherches commençant par le même texte. Enfin, l'historique peut être affiché visuellement en appuyant sur `[q]` avant d'entrer en mode de ligne de commande. Cela va créer une nouvelle fenêtre contenant l'historique en mode normal (excepté la touche `[Enter]` qui sert à exécuter la commande ou la recherche sous le curseur). On peut même faire des recherches dans cet historique, incluant l'historique des recherches ! Pour quitter l'historique sans rien exécuter, il suffit d'entrer `[ :q]`.

Le mode de commande pour objet de recherche peut être activé en appuyant sur `[/]` ou sur `[?]`. Dans le premier cas, la recherche est effectuée vers l'avant ; dans l'autre cas vers l'arrière.

Une fois entré en mode de commande avec `[/]` ou `[?]`, le texte à rechercher doit être entré. Lorsque c'est fait, il faut appuyer sur `[Enter]` pour démarrer la recherche.

La dernière recherche peut être répétée en n'entrant aucun texte après `[/]` ou `[?]`. On peut aussi utiliser la commande `[n]`, qui répète la dernière recherche dans la même direction, ou `[N]` pour rechercher dans la direction opposée.

Notons que toutes ces touches (`[/]`, `[?]`, `[n]`, `[N]`) servent à identifier un objet ; par le fait même, elles peuvent être précédées d'un compte et d'une commande. Ainsi, 2/xyz suivi de `[Enter]` va trouver la deuxième occurrence de xyz à partir du curseur et y placer ce dernier. 2d/xyz suivi de `[Enter]` va effacer jusqu'à la deuxième occurrence de xyz à partir du curseur.

Certains caractères sont spéciaux et doivent être précédés par une barre oblique inverse \ dans le texte de recherche. Ces caractères sont :

. \ * ^ $ [~

Ainsi, pour rechercher un astérisque, entrez `[/\*]`.

Ces caractères ont une signification spéciale car vi ne prend pas littéralement le texte que vous lui demandez de rechercher : il considère ce texte comme étant un motif de recherche. Ainsi, il est possible de faire des recherches plus générales. Ces recherches sont écrites dans un langage appelé **expressions régulières**. Expliquer complètement les expressions régulières prendrait un livre au complet ; la théorie sous-jacente est d'ailleurs vue au DIRO dans un cours de deuxième année de baccalauréat. Voici toutefois quelques exemples utiles de recherches employant des expressions régulières :

/abc	Cherche la chaîne abc.
/abc$	Cherche la chaîne abc en fin de ligne.
/^abc	Cherche la chaîne abc en début de ligne.
/$	Cherche une fin de ligne.
/^abc$	Cherche une ligne ne contenant que la chaîne abc.
/^$	Cherche une ligne vide.

/a.c	Cherche une chaîne dont le premier caractère est a, le second n'importe quoi et le troisième c. Pourrait trouver "abc", "azc", "a c".
/ba*c	Cherche une chaîne commençant par b, comportant des "a" et finissant par c. Pourrait trouver "bac", "baac", "baaaac" et même "bc".
/ba\+c	Cherche une chaîne commençant par b, comportant au moins un "a" et finissant par c. Pourrait trouver "bac', "baac" ou "baaaac".
/b\(ad\)*c	Cherche une chaîne commençant par b, comportant des "ad" et finissant par c. Pourrait trouver "badc", "badadc" et aussi "bc".
/\<bac	Chercher bac au début d'un mot.
/\<bac\>	Cherche bac formant un mot complet.
/b[ad]c	Cherche une chaîne dont le premier caractère est b, le second a ou d et le troisième c. Pourrait trouver "bac" et "bdc".
/b[ad]*c	Cherche une chaîne dont le premier caractère est b, une suite de a ou de d et qui se termine par c. Pourrait trouver "bc", "bac", "bdc", "baac", "badc", "bdac", "bddc", "baaac", etc.
/\<[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]	Cherche un numéro de téléphone local.
/\<\d\d\d-\d\d\d\d	Cherche un numéro de téléphone local.

Commandes spéciales

Les commandes suivantes n'emploient pas le format régulier du mode normal que nous avons vu ci-dessus.

F1	Démarre le système d'aide
:	Entre en mode de commandes ex
/	Entre en mode de recherche avant ex
?	Entre en mode de recherche arrière ex
Q	Quitte le mode normal et entre en mode de commandes ex continu
u	(undo) annule le dernier changement (peut être utilisé plusieurs fois)
U	(Undo) annule tous les changements effectués sur la ligne courante. Cette commande peut être annulée par `[u]`
v	(visual) passe en mode de sélection visuelle (voir plus bas)
V	(Visual) passe en mode de sélection visuelle de lignes (voir plus bas)
ZZ	(Zip) sauve le fichier courant s'il a été modifié et quitte.
ZQ	(ZipQuit) quitte immédiatement (comme :q!).
.	Répète la dernière commande
r	(replace) Remplace le caractère sous le curseur par le prochain caractère qui sera entré au clavier
J	(join) Fait suivre la ligne suivante directement à la fin de la ligne courante (sans retour de chariot)
~	change la casse du caractère courant (a ← A, A ← a).

Exemples

Voici quelques exemples de commandes.
[43G] se rend à la ligne 43 du fichier.
[G] se rend à la dernière ligne du fichier.
[2k] positionne le curseur deux lignes plus haut.
[45|] positionne le curseur à la colonne 45.
[3Ctrl-D] positionne le curseur trois demi-écrans plus bas.
[dw] efface le prochain mot.
[3dw] efface les trois prochains mots.
[d3w] efface les trois prochains mots.
[c3l] efface les trois prochains caractères et entre en mode d'insertion.
[cfx] efface jusqu'au prochain caractère x sur la ligne courante (sauf a) et entre en mode d'insertion.
[2cfx] efface jusqu'au second caractère x suivant sur la ligne courante (sauf ce dernier caractère) et entre en mode d'insertion.
[5x] efface les cinq prochains caractères.
[Y] copie la ligne courante dans le registre sans nom.
[10Y] copie la ligne courante et les 9 suivantes dans le registre sans nom.
["a10Y] copie la ligne courante et les 9 suivantes dans le registre a.
["A2yw] ajoute les 2 mots suivants au registre a.
["a2p] colle deux fois le contenu du registre a après la position actuelle du curseur (note : s'il y a au moins une ligne complète dans le registre, la copie commencera après la ligne courante).
[2i] insère deux fois le texte qui sera tapé avant la position courante du curseur.

Mode visuel

Le mode visuel est une exclusivité de VIM. Il permet de sélectionner une zone de texte pour ensuite lui appliquer une opération, un peu comme dans la plupart des éditeurs sous Windows. A partir du mode normal, il y a trois façons d'entrer en mode visuel :

- [v] pour une sélection caractère par caractère.
- [V] pour une sélection ligne par ligne.
- [Ctrl-V] pour une sélection rectangulaire.

Une fois dans le mode visuel, la sélection peut être étendue ou rétrécie en utilisant les touches référençant des objets, incluant les touches fléchées.

Lorsque vous avez terminé de choisir la zone où agir, entrez simplement la commande à exécuter (par exemple, [d] pour *delete*). La commande sera exécutée sur toute la région.

Deux touches s'emploient différemment en mode visuel : [u] sert à mettre la sélection en minuscules ([gu] en mode normal) ; [U] sert à mettre la sélection en majuscules ([gU] en mode normal).

Commandes du mode ex

On entre en mode ex en appuyant sur [:]. Le curseur sera alors dirigé vers la dernière ligne de l'écran et attendra une commande. Les commandes du mode ex prennent la forme d'un mot. Ce mot peut être suivi d'arguments et précédé d'une adresse. Ainsi, les commandes du mode ex prennent généralement la forme :

:[intervalle] commande [paramètres]

L'intervalle indique à quelles lignes s'applique la commande en cours. Ils ne sont pas utilisés pour toutes les commandes. En général, ils ont la forme *« début,fin »* ; par exemple : 3,7. Pour spécifier la ligne courante, employez *« . »*. Pour spécifier toutes les lignes, employez simplement *« % »*.

Obtenir de l'aide

La commande à utiliser en mode ex est :help. Entrée toute seule, la commande ouvre une nouvelle fenêtre.

Commande set

Une grande partie du comportement de l'éditeur dépend de sa configuration courante. La commande :set permet de changer ces options. Certaines options demandent un argument. Par exemple :

:set history=100

indique à vim de se rappeler des 100 dernières commandes entrées à la ligne de commande.

D'autres commandes sont booléennes (oui ou non) :

:set wrap

pour que le texte qui dépasse à la droite de l'écran soit affiché sur la ligne suivante de l'écran.

:set nowrap

pour que le texte qui dépasse à droite de l'écran ne soit pas affiché tant qu'on n'y accède pas avec le curseur.

On peut voir la valeur courante d'une option qui demande un argument en appuyant sur `[Tab]` après avoir entré l'option suivi du symbole =. Enfin, on peut voir toutes les options qui ne sont pas à leur valeur par défaut en entrant :set tout court. Pour voir toutes les options, entrez :set all.

Les paragraphes suivants décrivent quelques commandes utiles du mode ex.

Recherche et remplacement

Nous avons déjà vu la commande de recherche dans le mode normal. La même commande existe aussi dans le mode ex. Par contre, seul le mode ex possède une commande de remplacement. Cette commande est aussi utilisée par beaucoup d'autres programmes dans le monde UNIX, notamment ed, sed et perl.

:s/texte_a_rechercher/texte_a_remplacer/paramètres

Le *« s »* signifie *« substitute »*. Par exemple, pour rechercher le mot *« programme »* et le remplacer par le mot *« logiciel »*, entrez :

:s/programme/logiciel/

Par défaut, la commande n'affecte que la ligne courante et cesse dès que le premier remplacement est effectué. Pour que le remplacement s'effectue sur la ligne au complet, ajoutez le paramètre g. Pour qu'il s'effectue sur plusieurs lignes, spécifiez une adresse. Pour que le remplacement ne tienne pas compte de la casse, ajoutez le paramètre i.

Exemples :

:s/a/b/ remplace le premier a par b. Donc tata devient tbta.

:s/a/b/g remplace tous les a de la ligne par b. Donc tata devient tbtb.

:1,3s/a/b/g remplace tous les a des lignes 1 à 3 par b.

:1,.s/a/b/g remplace tous les a de la ligne 1 à la ligne courante par b.

:.,$s/a/b/g remplace tous les a de la ligne courante à la dernière ligne du texte par b.

:.-1,.+2s/a/b/g remplace tous les a de la ligne précédente à deux lignes plus loin que la ligne courante par b.

:.-1,.+2s/a/b remplace le premier a de chacune des lignes à partir de la ligne précédente à deux lignes plus loin que la ligne courante par b.

:%s/a/b/ remplace les premiers a de chaque ligne par des b dans tout le texte.

Évidemment, il est possible d'employer des expressions régulières. Voir à ce sujet la section sur la recherche en mode normal.

Deux options intéressantes existent pour contrôler la recherche (utilisez set pour changer ces options). La première option est **ignorecase** (ou **ic** en raccourci). Cette option fait que les recherches démarrées à l'aide de / ou :/ ignorent la casse.

La deuxième option est **incsearch** (ou **is** en raccourci). Cette option affecte les recherches effectues à l'aide de / ou ? en mode normal uniquement. Son activation fera en sorte de faire déplacer le texte pour voir la prochaine occurrence au fur et à mesure qu'elle est entrée au clavier.

Justification et marges

La marge est réglée par l'option **textwidth** ou **tw** en raccourci. Ainsi, pour régler une marge de 76 caractères (idéale pour la composition de courrier électronique), entrez :

:set tw=76

Le curseur va automatiquement revenir à la ligne après 76 caractères. Pour enlever la marge, entrez 0 comme largeur de texte. Pour reformater du texte déjà écrit à une nouvelle marge, utilisez en mode normal la commande `[gq]`.

On peut aussi cadrer à gauche, centrer et aligner à droite une (ou plusieurs) ligne(s) avec les commandes

:left :center :right

Mise en retrait automatique

Vi peut se charger de faire la mise en retrait automatiquement lorsque vous entrez des programmes C, C++ ou Java. Il suffit d'activer l'option **cindent**. (Pour d'autres langages, **smartindent** peut être utilisé. Pour les langages de la famille Lisp, **lisp** existe. Pour du texte normal, **autoindent** permet de garder les marges de gauches et les listes à point)

Lorsque cette option est activée et qu'on veut reformater une zone de texte après l'avoir tapée, on utilise la commande `[=]` avec l'objet désiré. Il existe `[==]` pour remettre en retrait la ligne courante.

En mode d'insertion, Ctrl-F permet la même chose.

Coloriage de la syntaxe

Le coloriage syntaxique peut être activé avec la commande

:syntax enable

Si vi ne devine pas correctement le type de syntaxe à employer, il est possible de lui indiquer avec l'option syntax :

:set syntax=java

Édition multiple

Vi peut éditer plusieurs fichiers à la fois. Cela est très utile pour transférer du texte d'un fichier à l'autre. VIM peut afficher plusieurs fichiers simultanément en utilisant des fenêtres. Mais, tout d'abord voyons comment utiliser le mode traditionnel où les fichiers sont affichés un à la fois.

Pour ce faire, il s'agit de mentionner sur la ligne de commande de vi les noms de tous les fichiers à éditer :

vi fichier1 fichier2 fichier3 ...

Vi va s'ouvrir avec le premier fichier. Une fois les opérations terminées, il suffit d'entrer la commande **:n** (*next*) pour passer au suivant. Dans VIM, pour passer directement à un fichier en particulier, il suffit d'entrer **:e #n** (*edit*) où n est le numéro séquentiel sur la ligne de commande. Par exemple, :e #3 passerait ici à fichier3.

Autrement, il est toujours possible d'ouvrir un fichier en entrant **:e nom-du-fichier**. Si le fichier courant contient des modifications qui n'ont pas été sauvegardées, vi va refuser l'opération. Pour l'effectuer quand même, entrez **:e !** au lieu de **:e**.

VIM possède un mode multifenêtres. Pour l'activer, on peut spécifier l'option **-o** sur la ligne de commande. Dans ce cas tous les fichiers spécifiés vont être ouverts dans leur propre fenêtre. A l'intérieur de VIM, il est aussi possible de créer une nouvelle fenêtre avec la commande **:split**. Employée seule, elle crée une nouvelle fenêtre avec le même fichier que la fenêtre en cours. Les changements effectués via l'une des fenêtres se répercutent automatiquement dans l'autre. Il est aussi possible de spécifier un nom de fichier après **:split**. Dans ce cas, la nouvelle fenêtre va s'ouvrir avec le fichier demandé.

La commande **:split** crée une barre de séparation horizontale. Pour créer une séparation verticale, employez simplement **:vsplit**. Enfin, pour créer une nouvelle fenêtre sans rien dedans, utilisez **:new** ou **:vnew**.

Pour se déplacer d'une fenêtre à l'autre, il suffit de taper `[Ctrl-W]` en mode normal, suivi d'une touche fléchée (ou de `[h]`, `[j]`, `[k]` et `[l]`). Pour fermer une fenêtre, on emploie la commande **:q**. Quand plusieurs fenêtres sont ouvertes et qu'on veut quitter l'éditeur, on peut employer **:qa** (*quit all*) au lieu de fermer les fenêtres une à une. Pour quitter et sauver toutes les fenêtres, il suffit d'entrer la commande **:wqa**. Enfin, pour quitter immédiatement sans rien sauver, il existe la commande **:qa !**.

Commandes du mode d'insertion

Nous avons déjà vu deux commandes du mode d'insertion : `[Esc]` quitte le mode d'insertion et retourne en mode normal. `[Ctrl-F]` met en retrait la ligne courante si l'option **cindent** est activée. Voici quelques autres commandes :

Ctrl-C Quitte le mode d'insertion.

Ctrl-H Efface le caractère sous le curseur.

Ctrl-K Permet de composer un caractère. Par exemple, Ctrl-K 'e entre é. Ctrl-K ^a entre â. Pour obtenir la liste des combinaisons possibles, utilisez la commande ex :digraph.

Ctrl-N Tente de compléter le mot courant à l'aide des mots semblables déjà dans le texte (recherche alphabétique avant).

Ctrl-V Permet d'entrer un caractère qui serait autrement intercepté par l'éditeur. Par exemple, pour entrer Ctrl-N dans le texte (^N), appuyez sur Ctrl-V Ctrl-N.

Ctrl-X Entre dans le sous-mode X. Voir l'aide pour les détails.

Ins Passe du mode d'insertion au mode d'écrasement et vice-versa.

Pour trouver toute autre information, référez-vous à l'aide dans **vi** (commande **:help** en mode normal). Toutes les informations y sont.

4.2.3 Pico

Introduction

Bien qu'en général on utilise les éditeurs plus puissants comme **Emacs** ou **VI**, il arrive qu'on veuille éditer un fichier rapidement et sans se casser la tête à se souvenir des commandes de ces deux derniers. **Pico** constitue un bon remplacement dans ces cas-là. Nous vous conseillons toutefois de lire la section de connexion à distance (section 6.3) pour trouver de meilleurs clients plutôt que de vous restreindre à l'usage de Pico.

Appel de pico

```
$ pico [fichier]
```

Interface

La première ligne à l'écran indique la version de Pico, le nom du fichier présentement ouvert et indique (par une astérisque) si le fichier a été modifié depuis la dernière sauvegarde. La troisième ligne en partant du bas affiche différentes informations relatives aux différentes commandes effectuées. Nous retrouvons aussi sur les deux dernières lignes un menu sommaire des différentes commandes disponibles.

FIG. 4.5 – Interface de Pico

Tout d'abord, il est utile de savoir que toutes les commandes disponibles avec Pico sont appelées avec l'aide de la touche `[ctrl]` suivie de la touche associée à la fonction. La notation `^x` signifie

[ctrl-x]. Cependant, il peut arriver que le programme de communication utilise déjà la séquence de contrôle. Si tel est le cas, l'usager pourra presser la touche d'échappement [esc] à deux reprises, pour ensuite presser la touche associée à la fonction, ce qui donnera le même résultat qu'avec la touche [ctrl]. Donc les commandes sont de deux formes : Soit ^x soit [esc] [esc] x.

Fonctionnalités

- ^g affiche l'aide.
- ^a déplace le curseur au début de la ligne.
- ^e déplace le curseur à la fin de la ligne.
- ^v déplace le curseur à la page suivante.
- ^y déplace le curseur à la page précédente.
- ^w effectue une recherche dans le texte.
- ^d efface le caractère qui est sous le curseur.
- ^k efface le texte marqué, sinon la ligne où se trouve le curseur.
- ^u récrit le texte effacé.
- ^j justifie un paragraphe.
- ^t appelle le vérificateur d'orthographe.
- ^c indique la position du curseur.
- ^r permet d'ouvrir un fichier et de l'insérer à la position du curseur.
- ^o permet de sauvegarder un fichier (avec confirmation).
- ^x quitte Pico et demande une confirmation de sauvegarde si le tampon a été modifié depuis la dernière sauvegarde.

Fonctionnalités avancées

Nous retrouvons avec Pico cinq tâches de base que peut effectuer l'éditeur. Nous avons la possibilité de justifier les paragraphes ; couper, copier et coller un bloc de texte ; effectuer une recherche dans le texte ; vérifier l'orthographe et utiliser un gestionnaire de fichiers.

Justification

La justification des paragraphes (^j) a lieu dans le paragraphe courant, c'est-à-dire dans le paragraphe où le curseur est situé. Si le curseur se trouve entre deux paragraphes, alors la justification a lieu sur le paragraphe sur la ligne suivante. Il est bon de noter que si nous voulons annuler la justification que nous venons d'effectuer, nous pouvons le faire à l'aide de la commande ^u.

Couper, Copier, Coller

Nous pouvons couper, copier et coller à l'aide des commandes ^_ (marque le début d'un bloc), ^k (efface) et ^u (annule l'effacement). La commande ^k efface tout le texte qui se trouve entre la marque et la position du curseur puis va placer le texte effacé dans un tampon. Si aucune marque n'a été mise, il effacera alors la ligne où se trouve le curseur. Avec la commande ^u, il sera possible de recoller le texte, se trouvant dans le tampon, à la position du curseur.

Recherche

Pour retrouver un mot dans le texte, nous pouvons utiliser la commande `^w`. Il faut noter que Pico, lors de la recherche, ne fait pas la différence entre les majuscules et les minuscules, et que la recherche s'effectue vers l'avant. La recherche bouclera le texte si elle s'avère infructueuse jusqu'à la fin.

Vérificateur d'orthographe

Si nous désirons vérifier l'orthographe des mots du texte, nous utilisons la commande `^t`. Il faut noter que, ici aussi, la vérification se fait vers l'avant et qu'il n'existe qu'un vérificateur de mots anglais.

4.2.4 Pilot : Gestionnaire de fichier

Lorsque nous chargeons `^r` ou sauvegardons `^o` un fichier, le gestionnaire de fichiers est accessible en tapant la commande `^t`. Ensuite, nous pouvons nous déplacer dans les différents répertoires à l'aide des flèches et des commandes de déplacement.

Ce gestionnaire de fichier est une version réduite du programme `pilot`, disponible aussi séparément à partir de la ligne de commande.

4.2.5 Autres

Des dizaines d'autres éditeurs de texte sont disponibles sur Linux : à vous d'expérimenter et de les essayer !

Mentionnons simplement `kedit` et `gedit`, deux favoris parmi la population étudiante.

4.3 Bureautique

Il est probable que pour certains de vos travaux et devoirs vous aurez à utiliser soit un logiciel de traitement de texte (en particulier pour les rapports), soit un chiffrier. Nous vous présentons ici quelques programmes qui remplissent, *grosso modo*, les fonctions que remplit sur la plateforme Windows la suite Office.

4.3.1 OpenOffice

La suite OpenOffice est un clône *Open-Source* de la suite Office de Microsoft. Elle permet d'ouvrir et de sauvegarder des fichier Office (Word, Excel, etc.) et ressemble beaucoup à Office. Elle inclut, entre autres, un traitement de texte, un chiffrier et un logiciel de dessin. Pour l'instant, c'est la version anglaise qui est installée.

Si vous voulez partir OpenOffice à partir d'une fenêtre de terminal, vous pouvez le démarrer avec la commande `ooffice` ou appelle directement un des sous composants evec les commandes `oowriter`, `oocalc`, `oodraw` ou `ooimpress`.

4.3.2 LaTeX

LaTeX est un logiciel de composition typographique adapté à la production de documents scientifiques et mathématiques de grande qualité typographique. Il permet également de produire toutes sortes d'autres documents, qu'il s'agisse de simples lettres ou de livres entiers. LaTeX utilise TeX comme outil de mise en page.

Nous conseillons fortement LaTeX pour l'édition de vos rapport. Des utilitaires permettent de générer des fichiers PDF, PS et HTML à partir du fichier LaTeX.

Le présent document à été réalisé avec LaTeX. Le code source est disponible sur la page Web des séminaires UNIX. Comme vous pouvez le constater, à partir de ce code source on peut générer la version PS, PDF ou HTML du manuel.

Tutoriel LaTeX
`http ://www.maths.tcd.ie/~dwilkins/LaTeXPrimer/`

URL

4.4 Archivage et compression

Il est souvent bien pratique de réduire la taille d'un fichier, que ce soit pour le télécharger plus rapidement, ou bien pour gagner de la place sur votre disque dur, ou encore de créer une archive de sauvegarde contenant plusieurs fichiers. Les outils de compression et d'archivage les plus employés sous UNIX sont `gzip` et `tar`.

Il est possible d'ouvrir les fichiers crées par **gzip** et **tar** avec **winzip** pour Windows.

Windows

4.4.1 gzip

Le programme affecté à la tâche de compression est **gzip** (GNU zip). Il s'agit en fait de deux commandes, `gzip` et `gunzip`.

Voici comment fonctionne **gzip** :

compression `$ gzip <fichier>`
Compresse **fichier** et le renomme **fichier.gz**.

décompression `$ gunzip <fichier.gz>`
Décompresse **fichier.gz** et le renomme **fichier**.

On ne peut pas employer **gzip** sur un répertoire

Attention

4.4.2 tar

`tar` est un archiveur très utilisé pour la distribution de logiciels et pour des tâches de sauvegardes. Il est capable de rassembler plusieurs fichiers et les combiner en un seul grand fichier pour les écrire

sur un périphérique de sauvegarde comme une bande.

Voici les options fréquemment utilisées :

-z –**gzip** –**gunzip**
demande à tar d'utiliser **gzip** pour compresser le fichier. Si vous utilisez cette option, il est bon d'ajouter l'extension **.gz** à votre fichier tar.

-c –**create**
demande à tar de créer une nouvelle archive.

-x –**extract**
demande à tar d'extraire une archive.

-v –**verbose**
active le mode verbeux : `tar` vous explique ce qu'il fait pendant qu'il crée l'archive.

-f <fichiers> –**file <fichiers>**
indique que la chaîne suivante de la ligne de commande est le nom du fichier à créer ou le périphérique à utiliser.

-t –**list**
Liste les fichiers contenus dans l'archive.

Notez bien que sans l'option **-z**, **tar** ne compresse pas les fichiers, il ne fait que les regrouper en un seul. Donner l'option **-z** est équivalent à appeler sans **-z** puis exécuter **gzip** sur le fichier obtenu.

L'extension standard d'un fichier créé avec **tar** est **.tar**. Si le fichier est compressé (que ce soit en donnant l'option **-z** ou en exécutant **gzip** sur le fichier .tar, l'extension standard est soit `.tar.gz` soit `.tgz` (les deux sont équivalents).

Voici les invocations de **tar** les plus fréquentes :

Créer une archive	`$ tar -cvf monfichier.tar <fichiers>*`
Extraire une archive	`$ tar -xvf monfichier.tar`
Créer une archive compressée	`$ tar -zcvf monfichier.tar.gz <fichiers>*`
Extraire une archive compressée	`$ tar -zxvf monfichier.tar.gz`

tar peut aussi utiliser la compression *« bzip2 »* qui est généralement plus puissante que *« gzip »* en donnant l'option **-j** à la place de **-z**. Il convient alors de donner `.tar.bz2` comme extension à votre fichier pour savoir en un clin d'œil qu'il s'agit d'une archive tar compressée par bzip2.

4.4.3 zip

unzip permet de compresser ou décompresser des fichiers **zip** (courants sur la plateforme Windows, malgré leur facteur de compression moindre).

Voici comment fonctionne **unzip** :

compression `$ zip <fichier.zip> <fichiers>*`
Compresse les **fichiers** dans **fichier.zip**
Les **fichiers** ne sont ni effacés ni renommés.

décompression `$ unzip <fichier.zip>`
Décompresse **fichier.zip** dans le répertoire présent.

`unzip` offre l'option **-l**, qui agit comme l'option **-t** de `tar`, c'est à dire que le contenu de l'archive sera affiché sans que l'archive ne soit extraite.

4.5 Imprimer

Cette section donne quelques exemples illustrant comment imprimer à partir des sources les plus courantes.

ATTENTION : il faut payer avant d'imprimer ! Consultez la section 5.2.

Assurez-vous d'avoir défini la variable d'environnement **PRINTER** tel que décrit dans la section 5.2

.

Attention

4.5.1 Imprimer une page web dans Firefox

Imprimer une page web en utilisant Mozilla ou Firefox est assez simple : dans le menu "File", choisir "Print". La boîte de dialogue d'impression vous apparaît (voir figure 4.6). Sélectionnez "Print to : printer" et donnez la commande `lpr`.

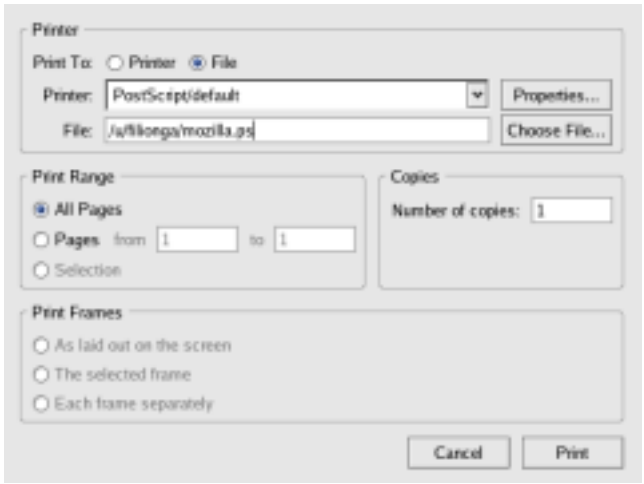


FIG. 4.6 – Sélectionnez "Printer" et donnez la commande lpr

4.5.2 Imprimer un fichier PDF avec Acrobat Reader

La procédure pour imprimer un fichier PDF dans Acrobat est à peu près la même que celle pour imprimer une page web avec Firefox. Dans le menu *« File »*, sélectionner *« Print... »*. Le dialogue d'impression vous apparaît (voir figure 4.7).

FIG. 4.7 – Sélectionnez “Printer” et donnez la commande lpr

4.5.3 Autres utilitaires

a2ps

Parfois, on peut désirer imprimer des fichiers en format texte seulement, du code source par exemple. **a2ps** convertit des fichiers textes vers le format PostScript (similaire au populaire PDF) pour fins d'impression et permet de sauvegarder le PostScript résultant dans un fichier pour visualisation ultérieure.

Les options les plus communes d'utilisations sont :

-o fichier.ps envoie la sortie dans le fichier **fichier.ps**

-PNOM imprime la sortie vers l'imprimante appelée **NOM**

-R orientation *« portrait »* (verticale)

-r orientation *« paysage »* (horizontale - par défaut)

-A permet d'imprimer deux fichiers différents sur une même page. Par défaut, si plusieurs fichiers sont spécifiés, on utilisera des pages différentes pour chacun des fichiers.

-C imprime le numéro de la ligne avant chaque 5 lignes (identique à **–line-numbers=5**)

–**columns=NUM** spécifie le nombre de colonnes par feuille. Par défaut, NUM = 2.

-B n'imprime pas l'en-tête habituelle

Par exemple, la commande suivante :

```
$ a2ps -A -o fichier.ps *.java
```

va convertir tous les fichiers du répertoire courant portant l'extension **.java** et va placer le résultat dans **fichier.ps** en format PostScript.

mpage

mpage permet d'économiser du papier en imprimant plusieurs pages sur une même feuille. **mpage** réduit la taille de chaque feuille et affiche le PostScript résultant dans la console. Si on veut sauvegarder le résultat de **mpage** dans un fichier, il nous suffit d'utiliser la redirection de la sortie vers un fichier.

Les principales options de **mpage** sont :

-2, **-4** ou **-8** imprime 2, 4 ou 8 pages par feuille (l'option -4 est mise par défaut)

-l Modifie l'orientation du papier de "portrait" vers "paysage"

-o Imprime ou non une bordure autour de chaque page réduite.

4.5.4 En cas de problème

Que faire si votre fichier ne s'imprime pas alors que vous avez lancé correctement la commande ? Ne relancez pas l'impression tout de suite : si le problème est temporaire vous risquez d'imprimer deux fois le même fichier ! Essayez plutôt un des trucs suivants.

1. Allez voir sur le petit écran de l'imprimante si un message d'erreur s'affiche. En général, si la lumière rouge clignote, quelque chose va mal !
2. Tapez la commande `lpq` et cherchez un message d'erreur qui décrirait la panne.

Si vous avez pu cerner le problème (par exemple : l'imprimante n'a plus de papier), ou encore si vous avez des problèmes qui ne proviennent pas de l'imprimante, contactez l'équipe du support technique au local 3195.

Gestion de la file d'impression

Les imprimantes du DIRO sont partagées entre tous les utilisateurs du DIRO à la fois. Lorsque vous imprimez votre document, il se peut qu'un autre utilisateur utilise déjà cette même imprimante. Votre impression n'est heureusement pas perdue et se retrouve dans une file d'attente avant d'être imprimée.

La commande **lpq** vous permet de voir le contenu de cette file d'attente :

```
$ lpq -Php3189
Printer: hp3189@papyrus  'HP 8000N publique local 3189'
 Queue: 3 printable jobs
```

```
 Server: pid 1928 active
 Unspooler: pid 1929 active
 Status: waiting for subserver to exit at 11:59:21.286
 Filter_status: getting end using 'pjl job/eoj' at 11:59:30.563
 Rank    Owner/ID                  Class Job  Files                 Size  Time
active  simardya@blg25+758          A   758  /tmp/AcroE8C9O1    2567158  11:58:13
2       jinf@contour+205            A   205  mozilla.ps        11554074  11:58:43
3       millema@blg23+481           A   481  document.ps        7542655  11:59:21
```

Le résultat de l'appel nous montre quel usager est actif sur l'imprimante, la taille des fichiers à imprimer ainsi que le numéro d'identification de chaque tâche d'impression. Supposons que l'usager millema désirer annuler l'impression de son document, pour ce faire, il suffit de lancer la commande **lprm** :

```
$ lprm 'millema*'
Printer hp3189@papyrus:
  checking perms 'millema@blg23+481'
  dequeued 'millema@blg23+481'
```

De cette façon, lprm annulera toutes les tâches lancées par l'usager millema. Notez les guillemets afin d'éviter que le shell ne fasse l'expansion des paramètres. Si plusieurs tâches peuvent satisfaire ce critère, on peut en annuler qu'une seule en notant le numéro de tâche et lancer la commande :

```
$ lprm 481
Printer hp3189@papyrus:
  checking perms 'millema@blg23+481'
  dequeued 'millema@blg23+481'
```

Un appel subséquent a `lpq` viendra nous assurer que notre document n'est plus dans la file d'impression :

```
$ lpq
 Printer: hp3189@papyrus  'HP 8000N publique local 3189'
 Queue: 2 printable jobs
 Server: pid 1928 active
 Unspooler: pid 1929 active
 Status: waiting for subserver to exit at 11:59:21.286
 Filter_status: getting end using 'pjl job/eoj' at 11:59:30.563
 Rank    Owner/ID                  Class Job  Files                 Size  Time
active  simardya@blg25+758          A   758  /tmp/AcroE8C9O1    2567158  11:58:13
2       jinf@contour+205            A   205  mozilla.ps        11554074  11:58:43
```

Il est à noter que les permissions vous permettent d'annuler les tâches vous appartenant, mais pas celles provenant d'autres usagers.

4.6 Commandes utiles

4.6.1 pager : less et more

Un *pager* est un programme qui permet de visualiser à l'écran des fichiers une page à la fois, un peu comme un éditeur de texte qui ne permettrait pas de modifier le fichier. Ils sont utilisés principalement par des programmes qui veulent faire lire quelque chose à l'usager ; vous pouvez aussi vous en servir pour visualiser rapidement le contenu d'un fichier.

`more` est le *pager* le plus classique. Il affiche le fichier page par page, avec une ligne au bas qui affiche *« More... »* et qui indique qu'on peut descendre plus bas dans le fichier (d'où son nom). Pour descendre d'une ligne, faites `[enter]` ; pour descendre d'une page, `[space]`. Notez que `more` ne permet pas de remonter dans le fichier - il le parcourt une seule fois. Pour visualiser un fichier, il suffit de taper

```
$ more nom-du-fichier
```

`less` est le successeur de `more` (d'où son nom). Pour le démarrer, il suffit d'entrer `less nom-du-fichier`. Contrairement à `more`, `less` permet de remonter et redescendre dans le fichier à volonté. Il permet aussi la recherche d'expression régulières. Tapez `h` pour avoir l'aide en ligne, `d` et `b` ou encore les flèches haut et bas pour faire défiler le texte et `/` suivi d'une expression régulière pour rechercher du texte selon cette expression (voir section 6.2.7).

Pour plus de détails sur `less`, consultez le manuel : `man less`.

Généralement, lorsque less est installé, la commande `man` utilise `less` pour faire afficher les pages d'aide. Apprendre à utiliser `less` sert donc en même temps à pouvoir utiliser le manuel de manière efficace.

Information

4.6.2 du

La commande `du` permet de voir l'espace qu'occupe sur le disque un répertoire. Elle est donc très pratique pour savoir combien d'espace vous utilisez dans votre compte (n'oubliez pas que vous êtes limité à 60 Mo).

Le manuel vous donnera tous les détails de son fonctionnement. Pour avoir un avant-goût, tapez la commande suivante dans votre répertoire de travail :

```
$ du -csh *
```

4.6.3 dos2unix, unix2dos et mac2unix

Quand vous transférez un fichier d'une machine Windows à une machine UNIX ou vice-versa, vous remarquerez peut-être des erreurs d'affichage à la fin des lignes. Cela est explicable par le fait que les deux systèmes d'exploitation ont différentes façons d'encoder les fins de ligne. Tandis que Windows considère `\r\n` comme étant la fin de ligne (une tradition héritée de son ancêtre DOS), UNIX utilise `\n` tout court. Pour compliquer les choses un peu plus, MacOS utilise `\r` seulement !

Les utilitaires `dos2unix`, `unix2dos` et `mac2dos` servent à convertir les fichiers d'un encodage à l'autre.

Pour réparer un fichier qui vient d'une machine Windows :

```
$ dos2unix le-fichier
```

Pour réparer un fichier qui vient d'une machine MacOS :

```
$ mac2unix le-fichier
```

Pour préparer un fichier à l'émigration vers Windows :

```
$ unix2dos le-fichier
```

Notez que ces trois commandes peuvent être remplacées par des appels à `perl` si vous vous trouvez sur un système qui ne les offre pas.

`dos2unix` est équivalent à
```
$ perl -i -pe 's/\r\n/\n/;' le-fichier
```

`mac2unix` est équivalent à
```
$ perl -i -pe 's/\r/\n/;' le-fichier
```

et `unix2dos` est équivalent à
```
$ perl -i -pe 's/\n/\r\n/;' le-fichier
```

on pourrait aussi définir un `unix2mac` avec
```
$ perl -i -pe 's/\n/\r/;' le-fichier
```

Bien sûr, ce n'est pas très lisible, mais cela fait le travail !

4.6.4 gimp

GIMP (GNU Image Manipulation Program) est un programme d'édition d'images très avancé et offrant un très grand nombre de fonctionnalités. Il s'apparente sur plusieurs points au populaire Photoshop de Adobe, disponible sur Windows et MacOS. Si vous avez besoin d'éditer une image, GIMP est le programme à utiliser.

Une explication de l'usage de GIMP dépasse la portée de ce document mais nous vous suggérons fortement d'aller lire le manuel en ligne :

URL

Guide utilisateur de GIMP
`http ://manual.gimp.org/fr/index.html`

4.7 Le compte casino

Selon la charte de l'AEIROUM :

> Le compte abrite des programmes autant utiles que ludiques qui sont disponibles à l'ensemble des usagers du réseau IRO. Il est à noter que la raison d'être du compte casino n'est pas de se substituer aux ressources systèmes (applications et cie) gérées par les administrateurs du réseau : toute ressource jugée essentielle pour la réalisation d'un cours (devoir, tp ou autre) sera entretenue par les administrateurs de IRO.

Le compte casino est donc un compte où les administrateurs élus par l'AEIROUM installent et maintiennent différrents logiciels pouvant être utiles aux utilisateurs du réseau de l'AEIROUM ou tout simplement ludiques.

Parmi les logiciels de casino, on compte des logiciels comme firefox, thunderbird, locus3, cowsay, quelques *window managers* altenatifs comme blackbox et fluxbox, quelques jeux comme atanks, xbomber. Cette liste est très loin d'être exhaustive et les logiciels contenus dans casino peuvent changer suite aux demandes qui sont faites aux administrateurs du compte (`casino@iro.umontreal.ca`).

Le compte casino maintient aussi un forum accessible au `http ://www-etud.iro.umontreal.ca/ casino/phpBB2/` .

Pour utiliser le casino (autre que le forum), il faut lancer la commande `source /u/casino/casino.bashrc` pour les utilisateurs de bash ou `source /u/casino/casino.cshrc` pour les utilisateurs de csh ou tcsh.

Chapitre 5

Le réseau du DIRO

5.1 Politiques d'utilisation

Ce module est consacré aux politiques d'utilisation des équipements du DIRO. Nous y verrons quels sont les droits et les devoirs des utilisateurs.

5.1.1 Les droits des utilisateurs

Les usagers ont certains droits et libertés, mais comme certains l'ont déjà dit avant : *« Ta liberté commence où se termine celle des autres »*. Il apparaît donc évident que certains droits auront plus de poids que d'autres. Par exemple, dans la société actuelle, le droit d'expression est subordonné devant le droit de respect de la vie privée du voisin. On ne peut pas aller publier les détails de sa vie sans son accord ! Il en va de même pour les usagers d'un réseau (UNIX ou autre). Les équipements du DIRO sont là pour permettre aux étudiants de l'Université d'y faire leurs devoirs et d'y étudier la matière vue en classe.

Le droit de travailler sur le parc d'équipements

Ce droit est évidemment fondamental. Il constitue la raison d'être du réseau. Lorsqu'un étudiant vient pour travailler (faire ses devoirs, étudier, etc.), il a donc priorité sur l'étudiant qui ne vient que pour *« jouer »* (que ce soit pour utiliser des programmes de jeux ou encore faire des choses qui n'ont qu'un lien très éloigné [lire aucun] avec l'étude). Ceci ne veut pas dire qu'une équipe a droit à un poste de travail pour chacun de ses membres, mais plutôt qu'elle a droit à au moins un poste de travail (bien sûr, cela dépend de ce que fait l'équipe).

Le droit de gestion de l'espace

Chaque usager se voit attribuer une certaine quantité d'espace sur les disques du département. Une fois cet espace alloué, l'usager a le droit d'en faire ce que bon lui semble. La façon dont un usager dispose de l'espace qu'il a (arborescence de sous-répertoires, noms de fichier, contenu) ne regarde que lui. S'il désire appeler son fichier *« Amadeus_Mozart »* au lieu de *« devoir_numéro_4 »*, c'est son droit (ou son problème, m'enfin ! ?). Il y a cependant des conventions que nous retrouvons chez plusieurs

utilisateurs et qui sont très fortement recommandés (comme se créer un répertoire `~/bin` où nous mettons les programmes *« faits maison »* à la disponibilité de tous, et `~/src` ou `~/bin/src` où nous plaçons les sources de `~/bin`). Ces standards permettent à tout le monde de mieux se comprendre et facilitent les échanges. Dans certains cas, ils permettent un meilleur fonctionnement du réseau (éviter les surcharges inutiles, car ça ralentit les processus de copies de sauvegarde). Mais tout cela demeure des conventions, et non des obligations.

Le droit à la gestion du temps

Certains réseaux UNIX n'offrent à leurs usagers qu'une durée limitée d'accès. Après un certain temps (quelques heures), l'usager est déconnecté et ne peut plus se brancher sur le réseau. Sur de tels réseaux, il importe donc de bien gérer son temps si nous voulons terminer nos travaux. Ce genre de pratique sert à dissuader l'utilisation des ressources à des fins personnelles. Rassurez-vous, il n'en va pas de même pour le réseau du DIRO. Cependant, dans les deux cas, ce que l'usager fait du temps qui lui appartient ne regarde que lui. S'il veut passer son temps à se promener sur le Web au lieu de faire son devoir de IFT 1010, pourquoi pas ? C'est lui qui sera pénalisé s'il est en retard. Bien sûr, nous pouvons faire ce que nous voulons tant que cela ne dérange personne. On ne parle pas de partir de grosses applications qui paralysent tout le réseau. Non plus que de jouer alors que quelqu'un attend qu'un poste se libère pour travailler !

Le droit à la confidentialité

Chaque usager a droit à ce que ce qu'il déclare comme privé (à l'aide des permissions ou autrement) le demeure. Même si certains usagers spéciaux ont la possibilité de passer outre les mécanismes de protection, il est clair que si un usager marque un fichier tel que lui seul peut y accéder, c'est que le contenu de ce ficher ne regarde que lui. De même, si le voisin tient ses choses privées hors de votre vue, c'est **très** probablement (lire sûrement) que cela n'a aucun intérêt pour vous !

Le droit à la libre utilisation des ressources disponibles

La meilleure censure que nous puissions appliquer à un logiciel, c'est tout simplement de ne pas le mettre sur le réseau, ou pour le moins de ne pas le rendre accessible. Nous pouvons donc en déduire que tout ce qui est sur le réseau est sans risque d'utilisation (comme disent les Américains : *« Guns don't kill, people do »*). Il ne faut pas hésiter à utiliser des logiciels publics tels 'tin' ou 'nethack'. S'ils étaient *« illégaux »*, ils ne seraient pas sur le réseau. Ce sont les principaux droits des utilisateurs, ceux auxquels se rapportent presque tous les autres auxquels nous pourrions penser. Ils sont simples, mais essentiels aux plaisirs de UNIX. Il ne faut jamais oublier que nous pouvons tenter plein de nouvelles expériences, en autant que nous ne nuisions à personne.

5.1.2 Les devoirs des utilisateurs

À la fin de ce séminaire, vous pouvez constater que le système UNIX est très *« puissant »* ; qu'il ouvre un univers d'applications aux utilisateurs. Il est toutefois important que ceux-ci prennent conscience non seulement des capacités du système et de leurs droits, mais aussi des devoirs envers les autres utilisateurs. Nous nous proposons, dans cette section, de vous indiquer les *« bonnes manières »* ; les choses à faire et à ne pas faire. Nous ne voulons pas être moralistes à outrance, mais quelques conseils ne sont jamais superflus !

Lorsque nous utilisons une ressource commune, il est important de respecter les autres utilisateurs et d'agir de façon responsable. Le présent texte se veut davantage un guide de *« l'écologie informatique »* ou, si vous préférez, un traité des *« bonnes manières »* plutôt qu'un règlement strict.

À faire et ne pas faire

Les machines et les logiciels de l'Université sont mis à la disposition des membres de la communauté universitaire pour des fins de recherche et d'enseignement. **L'utilisation de ces ressources pour des fins autres qu'universitaires, et en particulier leur utilisation dans le cadre d'un contrat ou d'un travail à l'extérieur de l'Université, est interdit**.

Les systèmes d'exploitation utilisés à l'Université encouragent le partage des informations et facilitent la communication. Les mécanismes de protection de l'information contre les accès non voulus, soit de l'intérieur, soit de l'extérieur du système, sont volontairement limités pour satisfaire deux objectifs incompatibles : assurer, d'une part, la confidentialité de certaines informations et maximiser, d'autre part, le partage des ressources. Les utilisateurs doivent donc suppléer aux carences du système d'exploitation et agir de façon à protéger les intérêts de la collectivité.

C'est ainsi qu'un utilisateur ne doit pas essayer d'avoir accès aux fichiers ni aux répertoires d'un autre utilisateur sans détenir au préalable l'autorisation de ce dernier. Cette autorisation peut toutefois être implicite : il suffit qu'un utilisateur rende ses fichiers accessibles aux membres d'un groupe ou du public. Nous ne devons pas non plus essayer d'intercepter une communication sur le réseau, qu'il s'agisse de courrier électronique ou d'un dialogue en direct, ou essayer d'avoir accès à toute information personnelle. Un programme public (un jeu, par exemple) ne doit pas être copié dans les fichiers personnels d'un utilisateur sans bonne raison.

Toute action entreprise délibérément par un utilisateur dans le but de modifier ou de porter atteinte à l'intégrité du système est illégale. De telles actions incluent l'utilisation non autorisée d'un compte ; les tentatives de se faire passer pour un autre utilisateur lors d'une communication ; d'apprendre des mots de passe ou de déchiffrer des informations encryptées ; et la destruction ou la modification de données ou de logiciels appartenant à d'autres utilisateurs. Toute tentative visant à empêcher les utilisateurs légitimes d'avoir accès au système ou à limiter leur accès est aussi à proscrire.

Le système inclut des logiciels et des données qui sont la propriété des utilisateurs ou d'une tierce personne, et qui sont protégés par des droits d'auteurs, des licences ou d'autres formes de contrat. Les utilisateurs doivent respecter ces restrictions, parmi lesquelles nous retrouvons : l'interdiction de copier de tels logiciels ou données ; de s'en servir à l'extérieur de l'université ; de les vendre ; de s'en servir pour des fins non universitaires ou pour réaliser un profit financier ; ou de les mettre à la disposition du public sans l'autorisation du propriétaire.

Nous espérons que vous serez de bons écologistes informatiques, et que vous nous aiderez à offrir les meilleurs services possibles, et à instaurer au sein de notre groupe une atmosphère de confiance et de respect réciproque.

Quelques conseils

Un petit coup de balai, ça fait du bien. N'hésitez donc pas à faire le ménage de vos fichiers de temps à autre, sinon les disques se remplissent comme par magie et tout le monde en souffre. N'oubliez pas les fichiers `.bak` ou `.bk` ! qui se créent automatiquement dès que vous éditez un fichier avec certains logiciels. De plus, les fichiers dont le nom commence par un point échappent plus souvent

qu'autrement à votre regard (utilisez `ls -a`). Finalement, ne conservez pas de copie personnelle de jeux ou de gros fichiers que vous pouvez trouver ailleurs, car en ce cas, la multiplication des pains ne sera rien comparée à la prolifération des copies multiples.

Il est très important que vous lisiez le courrier électronique qui vous est adressé. Vous y trouverez bien sûr des offres intéressantes avec, en prime, des messages importants de vos administrateurs favoris (enfin...). D'ailleurs, en faisant le ménage de votre fichier de messages personnels, mais aussi et surtout de votre boîte aux lettres système, vous diminuerez le temps de démarrage de chacune de vos sessions, tout en accélérant la livraison du courrier destiné aux listes d'usagers. Pensez-y !

La commande `chmp` sert à changer votre mot de passe. Il est important de bien choisir celui-ci et de le changer régulièrement. N'utilisez surtout pas de nom ou de prénom, ou de mot commun comme *« secret »*. Votre mot de passe est comme votre brosse à dent : vous l'utilisez fréquemment, vous le changez régulièrement et surtout vous ne le prêtez à personne. C'est vous, et vous seul, qui êtes responsable de votre compte. S'il devait se produire quelque chose de fâcheux, c'est vous qui seriez embêté, pas celui ou celle à qui vous auriez prêté votre compte.

Soyez prudent lorsque vous utilisez un programme inconnu ; celui-ci s'exécute avec vos permissions et non avec celles du propriétaire du programme.

Timeo Danaos, et dona ferentes.

5.2 Imprimer au DIRO

5.2.1 L'imprimante payante

Le DIRO met une imprimante laser payante à la disposition des étudiants du bacc. Cette imprimante se trouve dans le local 3189. Sur le réseau, on l'appelle **hp3189**.

Vous devez maintenant configurer quelle imprimante utiliser. La plupart des programmes permettent de spécifier l'imprimante au moment de l'impression. Toutefois, comme une seule imprimante est mise à votre disposition, nous vous recommandons d'indiquer l'imprimante dans votre fichier fichier d'initialisation de *shell*. Vous n'aurez à le faire qu'une fois, et toutes vos impressions seront faites sur la bonne imprimante.

Spécifier l'imprimante est très facile : il suffit de définir la variable d'environnement **PRINTER** en lui donnant la valeur *« hp3189 »*. Voyez la section 6.2.2 pour savoir comment faire.

5.2.2 Combien ça coûte, comment payer

L'impression au DIRO coûte huit sous par page. Vous devez payer avant de pouvoir imprimer.

Vous devez d'abord charger de l'argent sur votre carte d'étudiant. Plusieurs machines à cet effet se trouvent sur le campus (voir `http ://www.polycop.umontreal.ca/payants.html`) ; l'une de ces machines est situé dans la bibliothèque du pavillon André-Aisenstadt. Insérez-y votre carte de l'Université puis l'argent que vous désirez y déposer (des instructions plus détaillées se trouvent sur la machine elle-même). Si la bande magnétique de votre carte est abîmée et ne fonctionne pas correctement, vous pouvez acheter sur place une carte de polycopie qui remplit les mêmes fonctions.

Information

L'argent que vous déposez sur votre carte peut aussi servir aux autres services de polycopie de l'Université, par exemple aux photocopieuses.

Vous devez maintenant transférer l'argent de votre carte à votre compte. Une fois votre carte chargée d'un montant suffisant, rendez-vous au local 3189. À côté de l'imprimante se trouve un poste de travail servant uniquement au transfert d'argent de votre carte à votre compte. Branchez-vous avec votre nom d'usager et votre mot de passe habituels. Suivez les instructions qui apparaissent à l'écran pour transférer l'argent puis vous déconnecter. Si le montant que vous avez déposé dans votre compte est suffisant, vous pouvez maintenant imprimer à partir des postes de travail (voir 4.5).

Vous pouvez savoir en tout temps combien d'argent vous reste dans votre compte avec la commande `moncredit`. La commande affiche combien d'argent vous reste pour l'impression.

Pour plus détails concernant cette procédure ou pour des informations possiblement plus à jour que ce document, consultez ces pages web :

URL

Service de polycopie de l'Université
`http ://www.polycop.umontreal.ca/`

URL

Foire Aux Questions du DIRO, section Imprimantes
`http ://support.iro.umontreal.ca/faq_matimp.shtml#imprimantes`

5.3 Les machines

5.3.1 Stations de travail

Le DIRO met à la disposition de ses étudiants près d'une centaine de postes de travail. Ces postes de travail se trouvent tous au pavillon André-Aisenstadt.

5.3.2 Serveurs

On se branche, de l'extérieur, sur l'un ou l'autre des serveurs Linux qui sont dans le réseau du DIRO en utilisant le nom générique frontal.iro.umontreal.ca. Ce nom redirige vers un des serveurs frontaux du diro. (ex. : frontal01). Un autre serveur contient vos fichiers personnels et les *« exporte »* [1] aux serveurs frontaux et aux stations de travail. Vous n'avez pas accès à cette machine directement. Vous disposez de 60 Mo d'espace disque pour vos fichiers personnels.

5.4 Les commandes spécifiques au DIRO

5.4.1 remise

Dans le cadre de plusieurs cours, vos devoirs devront être remis sous forme de fichiers (code source d'un programme, par exemple). La remise de ces devoirs se fait via la commande `remise`, qui soumet les fichiers pour vous et vous permet de voir la liste des fichiers que vous avez déjà remis. L'usage va comme suit :

Pour une remise

[1] par NFS, pour les curieux

```
$ remise <cours> <travail> <liste des fichiers>
```

Pour une vérification

```
$ remise -v <cours> <travail>
```

Paramètres

-v	Quand cette option n'est pas spécifiée, la commande sert à remettre les fichiers. Avec cette option, elle sert à voir la liste des fichiers déjà remis.
<**cours**>	Sigle du cours (exemple : ift1165).
<**travail**>	Nom du travail (exemple : tp1). Ce nom est déterminé par le professeur ou les démonstrateurs qui vous le communiqueront, généralement dans l'énoncé du travail.
<**fichier**>	Nom du fichier à remettre (exemple : prog.c). On peut fournir une liste de fichiers à remettre (exemple : *.h *.c).

Souvent, le professeur ou les démonstrateurs vous donneront la liste exacte des fichiers que vous devez remettre pour un certain travail, avec la ligne de commande exacte à utiliser.

5.4.2 notes

`notes` permet à chaque étudiant d'avoir accès à ses notes (résultats aux travaux et aux examens) pour un cours qui a un compte diftxxxx sur le réseau. Le programme vous permet seulement d'avoir accès à vos propres notes, pas celles des autres.

Par exemple, pour lire ses notes des cours ift1214, l'étudiant doit être connecté dans son compte, et y taper la commande

```
$ notes ift1214
```

Il est à noter que certains professeurs préfèrent ne pas publier les notes via le réseau, et que dans ces cas vous aurez un message du genre *« Ce cours n'existe pas »*. Ces professeurs vous indiqueront comment obtenir vos notes.

5.4.3 inclure

`inclure` permet de configurer votre *shell* présent pour l'utilisation d'un logiciel. Pour obtenir la liste des logiciels gérés par le programme **inclure** tapez :

```
$ inclure
```

Pour inclure le compilateur Java **jdk-1.5**, par exemple, tapez :

```
$ inclure jdk-1.5
```

Le programme inclure n'affecte que le *shell* courant. Il est donc possible de configurer le compilateur **jdk-1.1** et **jdk-1.3** dans deux *shell* différents. Pour rendre permanente l'inclusion d'un logiciel, ajoutez la commande dans votre fichier d'initialisation de *shell* (voir la section 6.2.1).

5.5 Les adresses électroniques et Web des cours

Pour chaque cours donné au DIRO (cours dont le sigle est IFT), deux paires composées d'une adresse de courrier électronique et d'une page web sont créées : une pour les démonstrateurs du cours et une pour le professeur. Une liste de diffusion pour tout le groupe est aussi mise à jour.

Prenons l'exemple du cours **IFT1010** :

dift L'usager **dift** représente les démonstrateurs. Les messages envoyés à dift1010 seront lus par les démonstrateurs du cours. La page web dift1010 est mise à jour par les démonstrateurs.

pift L'usager **pift** représente le professeur. Les messages envoyés à pift1010 seront lus par le professeur du cours. La page web pift1010 est mise à jour par le professeur.

gift La liste **gift** représente tout le groupe. Les messages envoyés à gift1010 seront reçus par tout le groupe (n'envoyez pas de message à ce groupe sans raison valide : vous risquez de vous faire couper votre compte en punition).

Chapitre 6

Pour en savoir plus

6.1 Historique de UNIX

Dans cette section nous voulons donner un bref aperçu de l'historique de UNIX et du système d'exploitation Linux.

6.1.1 UNIX

UNIX désigne une famille de systèmes d'exploitation dont le premier a été conçu aux laboratoires Bell (Bell Laboratories). C'est un système qui est assez vieux (une bonne trentaine d'années), utilisé tant pour les gros ordinateurs que pour les plus petits. Nous le retrouvons sur les super-ordinateurs (Cray), sur les ordinateurs centraux, sur les minis (VAX, HP), sur les postes de travail (HP, Apollo, Sun, SGI, ...) et bien sûr, sur les micros (Linux).

1960+ Il y a beaucoup de débats sur les mérites des différents langages de programmation (PL/1, APL, Simula, ALGOL 60, COBOL, FORTRAN, etc.). À l'Université de Londres et de Cambridge, le langage BCPL (Basic Combined Programming Language) est créé. Des recherches sur les concepts de temps partagé, de traitement interactif (par opposition au traitement par lots), de pagination et de protection de la mémoire, d'ordonnancement des travaux et de structure de fichiers sont débutées.

1967 Dennis Ritchie quitte Harvard pour travailler aux Laboratoires Bell dans le New Jersey.

1968 Ken Thompson quitte Berkeley, où se faisaient déjà des recherches sur un nouveau système d'exploitation (SDS930), pour se joindre à une équipe de spécialistes qui avaient travaillé sur les systèmes Multics (Cambridge Multiple Access System) et GE 645.

1969 Thompson et Ritchie produisent la **première édition** d'un système à usager unique sur un PDP 7. C'est un système primitif qui ne comporte qu'un assembleur et un chargeur.

1970 La primitive *« fork »* est ajoutée pour permettre la création de processus et des programmes utilitaires pour la gestion des fichiers sont produits (**deuxième édition**). Le système accepte alors deux usagers et Thompson le baptise UNIX. Ken Kernighan

avait, un jour, fait référence au système par : *« Uniplexed Information and Computing System »* (UNICS) par opposition à MULTICS (Multiplexed Information and Computing System).

1971 Le système est transporté sur un PDP 11/20 et un système de traitement de textes (roff) est produit pour le service des brevets des Laboratoires Bell (le premier utilisateur extérieur de UNIX). Thompson et Ritchie publient la première documentation du système. C'est la **troisième édition**.

1972 UNIX est amélioré en lui ajoutant la notion de relais (*pipe*). Thompson développe le langage B (un descendant de BCPL) et réécrit l'assembleur du système en B. C'est un compilateur qui produit du code interprétable, peu performant et Ritchie produit un générateur de code pour le PDP 11. C'est le langage C. Il existe à l'époque environ 20 sites utilisant le système UNIX.

1973 UNIX est réécrit en langage C (**quatrième édition**). Il fonctionne alors sur un ordinateur avec un disque rigide de 500K et peut supporter environ cinq usagers. Thompson se charge de la gestion de processus tandis que Ritchie s'occupe de la gestion des entrées/sorties.

1974 Plusieurs universités commencent à utiliser UNIX et la **cinquième édition**, conçue spécialement pour des fins académiques, est introduite.

1975 La **sixième édition** de UNIX est produite et devient la première à être commercialisée pour une somme modique par AT&T.

1977 Près de 500 sites utilisent UNIX. Une nouvelle version pour un ordinateur Interdata 8/32 est produite. Le langage C est aussi amélioré. John Reiser et Tom London, des Laboratoires Bell, écriront UNIX 32V pour le VAX 11/780. C'est un descendant de cette version qui est actuellement distribué par l'Université de Berkeley en Californie (UCB).

À partir de ce moment, les versions de UNIX vont se multiplier ; BSD4.2 de Berkeley, A/UX de Apple Computer, HP-UX de Hewlett-Packard, AIX de IBM, Domain/IX de Apollo Computer et SunOS de Sun Microsystems. C'est à cette époque que débute le problème de standard.

1980 Il y a environ 100 000 sites UNIX ! Le problème, c'est que personne ne supporte vraiment ce système. Les détenteurs de licences sont laissés à eux-mêmes.

1983 AT&T annonce **System V** qui est complètement différent de la version de Berkeley.

1984 La cour des États-Unis brise le monopole de AT&T et l'oblige à se subdiviser. Sept compagnies naîtront (les Baby Bells) et AT&T aura le droit de vendre des produits informatiques (de la concurrence pour IBM). Elle décide de miser sur UNIX, produit la **version 2 de System V** et offre un support complet pour le système d'exploitation et les utilitaires.

1986 AT&T annonce la **version 3 de System V** qui supporte RFS (Remote File Sharing).

1987 Une nouvelle version de UNIX compatible avec la version de AT&T, celle de Berkeley, de Sun et XENIX de Microsoft est annoncée. C'est **System V version 4.0** qui sera disponible à l'automne 1989. Donc deux versions vont subsister : **System V 5.3** de AT&T et **BSD 4.3** de Berkeley.

1988 Un mouvement se forme pour produire une version compétitrice de UNIX. C'est **OSF** (Open Software Foundation) qui regroupe : IBM, Hewlett-Packard, BULL, SIEMENS et Apollo.

1989 Au départ, toutes les sources de UNIX étaient disponibles, donc beaucoup de gens ont pu contribuer à son expansion. Avec la commercialisation du système par AT&T, les sources n'étaient offertes qu'à des prix exorbitants. Le projet **GNU** (GNU's not UNIX) a pour objectif (entre autres) de remettre UNIX dans le domaine public. Sur une base de volontariat, les participants au projet GNU produisent du code UNIX disponibles gratuitement.

6.1.2 Linux

1991 Linus Torvalds, étudiant à l'université d'Helsinki (Finlande) installe le système Minix sur son i386. Le système s'avérant trop limité pour Linus, il décide d'aller plus loin sur la base de ce qui existe... **Linux** (Linus' UNIX, l'UNIX de Linus) 0.0.1 est né au mois d'août 1991. Linus lance alors un appel à contribution, et permet donc un libre accès au code source.

Cette version permet de faire tourner quelques applications GNU (logiciels libres) essentielles comme le compilateur **gcc** ou le shell **bash**. Linus prend la décision de mettre le code source sous licence GPL : tout le monde peut alors participer au développement de Linux.

1991-94 Le développement anarchique de Linux ne lui permet pas, au départ, de devenir un système compétitif face aux autres systèmes du marché. Son point faible réside dans son système de fichiers hérité de Minix. Seule l'intégration à Linux du Second Extended Filesystem (ext2fs), conçu par Rémi Card à partir du système de fichiers de BSD, permet d'en faire un système fiable. Ext2fs offre enfin les performances et les services de systèmes de fichiers professionnels. Il devient naturellement un standard.

La présence dans le processus de développement de développeurs venant de nombreux horizons permet à Linux d'exister sur de nombreuses plates-formes (Mac, Atari, Amiga, Alpha) autres que sa plate-forme d'origine (Intel). Petit à petit, Linux devient un système UNIX complet compatible avec les autres systèmes UNIX, offrant toujours plus de services de qualité professionnelle au plus grand nombre.

1995 L'explosion d'Internet donne un second souffle au développement de Linux : non seulement pour permettre à la communauté des développeurs de Linux de continuer à s'étendre, mais aussi et surtout pour donner à Linux une existence réelle sur le marché des systèmes d'exploitation. Ainsi, ses qualités d'OS libre (et gratuit), robuste et performant font qu'il est choisi par de plus en plus de fournisseurs d'accès à Internet. Il est ainsi devenu aujourd'hui le leader sur le marché de l'hébergement de sites Web.

Parallèlement, l'apparition et le développement de sociétés privées telles que **RedHat**, **Caldera** ou **VA Linux** donne une envergure jusqu'alors inconnue à Linux : les distributions deviennent de plus en plus conviviales et simples à installer, et des services professionnels sont mis en place pour faciliter l'implantation de Linux dans les entreprises.

1996 Linux commence à faire parler de lui dans les média. Red Hat Linux est élu meilleur

OS par InfoWorld. Début du projet **KDE** : on commence à développer des projets conviviaux pour le grand public.

1999 Linux est présenté comme une alternative au système Windows de Microsoft dans le domaine des serveurs. Linux est présent sur 35% des serveurs d'entreprises. Les salons Linux se multiplient et l'on observe une multiplication des revues spécialisées.

6.2 Concepts avancés de UNIX

6.2.1 Les interpréteurs de commandes

L'interpréteur de commandes est le programme avec lequel interagit un utilisateur lorsqu'il débute une session en mode terminal sur UNIX. Au fil des ans, plusieurs interpréteurs ont été développés. Le premier d'entre tous fut le Bourne Shell (sh) développé par Stephen R. Bourne aux Bells Labs. Plus tard, le C Shell (csh) fut développé par Bill Joy à l'Université de Californie à Berkeley. Il est nommé ainsi parce que sa syntaxe ressemble à celle du langage C. Une version améliorée fut ensuite développée, tcsh. Le projet GNU a ensuite développé son propre shell, fondé sur les normes d'interopérabilité POSIX. Cet interpréteur fut appelé le Bourne Again Shell (bash). C'est ce dernier qui est invoqué par défaut sur le réseau du DIRO . Nous parlerons toutefois aussi d'un autre interpréteur populaire, tcsh, utilisé en outre les années précédentes au département.

Afin d'alléger le texte, par la suite, nous emploierons le terme anglais *shell* pour désigner un interpréteur de commandes.

Les commandes

La section 3.5.1 donne un aperçu général des commandes interprétées par le *shell*. En guise d'un bref rappel, une commande est formée en premier lieu du nom du programme à exécuter, suivi possiblement d'options et de paramètres.

Plus en détail, le premier terme de la commande peut être le nom d'une commande externe (un programme ou un *script*), le nom d'une commande interne (ou *builtin command*), ou un alias (voir plus loin dans cette section). Dans le cas d'une commande externe, par exemple la simple commande suivante :

```
$ ls
```

le *shell* cherche un fichier portant le nom de la commande dans une liste de répertoires spécifiée par la variable d'environnement `PATH` (voir 6.2.2), dans l'ordre qu'ils apparaissent dans cette variable. Il faut bien entendu que ce fichier soit bel et bien une commande et qu'il ait les permissions d'exécutions appropriées (voir 3.4.3). Cette recherche n'est pas effectuée si le nom de la commande est spécifié de façon relative ou absolue (voir 3.4.2). Voici un exemple :

```
$ /bin/ls
```

Ici, le *shell* cherche directement à exécuter la commande qui se trouve dans le répertoire `/bin`. Il est parfois pratique d'employer la notation absolue ou relative s'il y a plusieurs commandes qui portent le même nom, ou pour exécuter un programme que vous venez de compiler pour votre travail

pratique. Par exemple, typiquement, le programme compilé se trouvera dans le répertoire courant. S'il s'appelle `toto`, vous l'exécutez alors ainsi :

```
$ ./toto
```

Le `./` indique de chercher `toto` dans le répertoire courant.

Dans le cas d'une commande interne, on ne spécifie que son nom (sans chemin relatif ou absolu, car dans un tel cas, le *shell* va automatiquement chercher une commande externe). Les *shells* `bash` et `tcsh` reconnaissent plusieurs commandes internes, dont voici les plus fréquemment employées (nous reparlerons plus en détail de plusieurs de ces commandes) :

`alias` Sans argument, la commande affiche tous les alias. Avec un argument, elle affiche l'alias pour cette argument. Avec deux arguments ou plus, la commande définit un l'alias portant le nom du premier argument et dont la *« valeur »* est le reste des arguments. Par exemple, si vous utilisez fréquemment la commande `ls -l`, vous pouvez vous définir l'alias `ll` (par exemple) avec la commande suivante :

`bash` : `$ alias ll="ls -l"` `tcsh` : `% alias ll ls -l`

Ainsi, par la suite, vous n'avez qu'à entrer la commande `ll` pour lister les fichiers avec leurs détails.

`bg` Met une tâche en arrière-plan (voir **Contrôle des tâches** plus loin).

`cd` Change le répertoire courant (voir 3.5.3).

`echo` Affiche ses paramètres (voir 3.5.11). L'option `-n` indique de ne pas terminer la ligne avec le caractère de fin de ligne.

`exit` Quitte le *shell*. Dans une fenêtre de terminal, ceci a pour effet de fermer la fenêtre. Si vous êtes connectés par `ssh` (voir 4.1.4) ou tout autre programme de communication similaire, la commande a pour effet de terminer la connexion.

`export` Cette commande est spécifique à `bash` et sert à transformer une variable de shell, créée avec **set** (voir plus loin) en variable d'environnement. Elle prend en paramètre le nom de la variable à exporter. Pour `tcsh`, un utilise `setenv`.

`help` Cette commande est spécifique à `bash`. Sans paramètre, elle donne la liste des commandes internes du *shell*. Suivie du nom d'une de ces commandes, elle en donne la description.

`fg` Met une tâche en avant-plan (voir **Contrôle des tâches** plus loin).

`history` Affiche la liste des dernières commandes effectuées. Si un nombre **n** est donné en paramètre, la liste donnera les **n** derniers événements.

`jobs` Affiche la liste des tâches courantes (voir **Contrôle des tâches** plus loin).

`kill` Tue une tâche ou un processus (voir **Contrôle des tâches** plus loin).

`logout` Similaire à `exit`, mais ne fonctionne que pour les instances de *shell* de type *login* (par exemple, si vous êtes branchés avec un programme de communication tel `ssh`).

`pushd`, `popd` Lorsque vous vous promenez de répertoire en répertoire fréquemment, il peut s'avérer intéressant de maintenir une forme d'historique des répertoires visités. Ainsi, au lieu d'appeler la commande `cd`, vous employez la commande `pushd`. Cette dernière vous amènera dans le répertoire spécifié en plus de se souvenir du répertoire précédent.

Pour revenir au répertoire précédent, vous employez la commande `popd`. A priori, ceci fait la même chose qu'employer la commande `cd` puis `cd` avec l'option `-` (voir 3.5.3), mais par contre avec `pushd` et `popd`, vous n'êtes pas limité à un seul niveau de récursion. Si la commande `pushd` est appelée sans argument, elle a pour effet d'interchanger la position du répertoire courant et précédent dans l'historique. À chaque appel de `pushd`, l'historique courant des répertoires est affiché. Voici un exemple (en supposant que vous être l'usager `semunix` et que le répertoire d'origine est le répertoire racine du compte) :

```
$ pushd toto
~/toto ~
$ pushd ../tata/sousrep
~/tata/sousrep ~/toto ~
$ pwd
/u/semunix/tata/sousrep
$ pushd
~/toto ~/tata/sousrep ~
$ pwd
/u/semunix/toto
$ popd
~/tata/sousrep ~
$ pwd
/u/semunix/tata/sousrep
$ popd
~
$ pwd
/u/semunix
```

`rehash` Cette commande est spécifique à `tcsh`. Reconstruit la table d'adressage interne des commandes. Dans `bash`, l'équivalent le plus proche est hash -r.

`set` Cette commande est présente autant dans `bash` que dans `tcsh`. Nous allons montrer son utilisation pour `tcsh`. En bref, cette commande permet de créer ou de changer la valeur d'une variable. Voici deux exemples simples :

`set toto` crée une variable `toto` avec un contenu vide ;

`set toto = allo` crée une variable (ou change son contenu si elle existait déjà) `toto` avec `allo` comme contenu ;

Sans argument, la commande affiche toutes les variables créées ainsi que leur contenu. Pour `bash`, il suffit de faire une assignation sans aucune commande :

`$ toto=` crée une variable `toto` avec un contenu vide ;

`$ toto=allo` crée une variable (ou change son contenu si elle existait déjà) `toto` avec `allo` comme contenu ;

Profitons de l'occasion pour indiquer comment accéder au contenu d'une variable. C'est simple, il suffit de préfixer le nom de la variable d'un signe de dollar. Par exemple, la commande suivante :

`echo $toto`

affiche le contenu de la variable `toto`. En cas de risque de confusion, on peut aussi ajouter des accolades autour du nom de la variable :

`$ echo ${totoa}` imprime le contenu de la variable totoa ;

`$ echo ${toto}a` imprime le contenu de la variable toto suivi de la lettre a ;

Il faut bien comprendre que ce n'est pas `echo` qui interpréte le nom de la variable, mais bien le *shell*, avant que `echo` n'entre en jeu. Ce que `echo` recevra comme argument, c'est simplement la valeur de la variable `$toto`, exactement comme si on avait tapé la valeur de la variable directement.

`setenv` Cette commande est spécifique à `tcsh`. Sans argument, elle a le même effet que `printenv`. Sinon, elle attribue une valeur à une variable d'environnement (voir 6.2.2). Pour `bash`, on utilise `export`, vue plus haut.

`source` Cette commande exige un argument, qui doit correspondre à un fichier texte. Elle exécute les commandes contenues dans le fichier, à raison d'une commande par ligne. Pour `bash`, une commande équivalente est simplement `.` (le point), sauf que le fichier doit aussi être exécutable (voir droit d'accès section 3.4.3

`stop` Cette commande spécifique à `tcsh` met une tâche en suspens (voir **Contrôle des tâches** plus loin).

`time` Il est souvent pratique de connaître le temps d'exécution d'une tâche. Pour ceci, il suffit de préfixer la commande par `time` et lorsqu'elle se terminera, des statistiques sur le temps d'exécution seront affichées. Exemple (dans `bash`, puis dans `tcsh`) :

```
$ time cp toto tata
real    0m0.010s
user    0m0.000s
sys     0m0.000s
```

Le premier nombre est le temps total (absolu, comme si vous aviez mesuré le temps entre le début et la fin de l'exécution du programme avec une montre). La somme des deux autres indique le temps total CPU qu'a pris la commande (si plusieurs programmes s'exécutent en même temps sur la machine, ce temps ne tiens pas compte de la portion prise par les autres programmes).

```
$ time cp toto tata
0.000u 0.020s 0:00.30 6.6\%     0+0k 0+0io 121pf+0w
```

Pour `tcsh`, la somme des deux premiers nombres affichés donne le temps CPU total (en secondes) qu'a pris votre commande. Le troisième nombre est le temps total absolu. Consultez les pages de manuel pour en savoir plus sur les autres nombres et comment changer le format d'affichage.

`umask` Par défaut, lorsque vous créez un nouveau fichier, il n'y a aucune permission d'aucun type donnée au groupe ou aux autres (voir 3.4.3). En général, c'est mieux ainsi, mais parfois on voudrait changer ça, en particulier lorsqu'on travaille dans un répertoire public (i.e. dont on veut que tout le monde ait accès). Comme c'est là, il faudrait manuellement changer les permissions, ce qui n'est pas nécessairement une mauvaise chose (en particulier si on ne veut pas rendre disponible un fichier sur lequel on est en train de travailler). Dans d'autres cas, par exemple si on veut extraire une archive (un `.tar.gz`) dans un endroit public de son compte, c'est moins intéressant. La commande `umask` permet de changer ça. Sans argument, la commande donne la valeur actuelle du masque.

Le masque est exprimé d'une manière similaire à la manière absolue de donner des permissions à un fichier (voir 3.5.10) sauf qu'au lieu de faire la somme des permission qu'on veut donner, ont fait la somme de celles qu'on ne veut pas donner. Exemple :

```
$ umask
77
```

Ceci veut dire que les nouveaux fichiers n'auront aucune permission d'aucune forme pour le groupe et les autres. Ce masque est équivalent à `077`. Le premier zéro indique qu'il n'y a pas de restriction sur les permissions de l'usager. Il est très rare de vouloir mettre autre chose qu'un zéro ici. Pour changer ce masque, il suffit de passer un argument de la même forme. Ainsi, si vous exécuter la commande suivante :

```
$ umask 022
```

ou alors la commande équivalente suivante :

```
$ umask 22
```

alors tous les fichiers ou répertoires qui seront créés pourront avoir les permissions de lecture et d'exécution pour le groupe et les autres, mais jamais d'écriture.

Attention

Soyez prudent avec l'usage de cette commande. Si vous désirez changer temporairement le masque pour effectuer une tâche, n'oubliez pas de le changer à sa valeur précédente après (i.e. `umask 077`).

`unalias` Cette commande fait le contraire de `alias`, i.e. elle détruit les alias donnés en argument. Par exemple, si vous voulez détruire l'alias `ll` (de l'exemple pour la commande `alias`), vous n'avez qu'à exécuter la commande suivante :

```
$ unalias ll
```

`unset` Cette commande détruit toutes les variables données en argument.

`unsetenv` Similaire à `unset`, mais pour les variables d'environnement (6.2.2). Pour `bash`, la commande est `unexport`. Cette commande enlève la variable de l'environnement, mais pour la détruire complètement, il faut utiliser `unset`.

`where` Cette commande est spécifique à `tcsh`. Si jamais il existe des conflits au niveau des noms des commandes (par exemple, il y a une commande qui s'appelle `toto` qui se trouve dans plusieurs répertoires du `PATH` (voir 6.2.2), et si on avait défini un alias du même nom), cette commande liste toutes les possibilités.

`which` Cette commande est spécifique à `tcsh`. Parmi les choix donnés par la commande `where` ci-dessus, cette commande indique quelle commande serait exécutée en employant ce nom.

Il existe une commande non interne `which` qui fait la même chose que celle-ci, mais qui ne considère pas les commandes internes. Cette commande est appelée lorsqu'on exécute which de `bash`. Dans `bash`, la bonne commande à appeler est `type`.

Attention

Enfin, le dernier type de commande : les alias (nous avons déjà vu la commande du même nom pour les créer).

`man bash` définit les alias comme suit : *« Les alias permettent de substituer une chaine à un mot lorsqu'il est utilisé comme premier mot d'une command simple. »* En français, cela signifie que si le premier mot d'une commande est défini en tant qu'alias, le shell fera une substitution pas une chaîne. Par exemple, si on veut une commande qui nous affiche tous les fichiers d'un répertoire.

```
$ alias la='ls -a'
$ la
.
..
.bashrc
Desktop
ns_imap
nsmail
```

Comme mentionné pour la commande interne `where`, il peut y avoir plus d'une commande qui porte le même nom. S'il s'agit d'une commande externe, le choix se fait comme nous l'avions mentionné quelques paragraphes plus haut (recherche en ordre dans le `PATH` ou sans ambiguïté avec un chemin relatif ou absolu). Mais si le conflit se produit au niveau de plus d'un type de commande, le *shell* regarde d'abord s'il y a un alias qui porte le nom. Sinon, il regarde s'il y a une commande interne du nom. Sinon, il démarre la recherche d'une commande externe.

Le traitement des options et des paramètres dépend bien entendu de la commande employée. Généralement, les options sont un ou plusieurs caractères préfixés d'un tiret, par exemple l'option `-l` de la commande `ls`. Une bonne part des commandes reconnaissent l'option spéciale `--`, qui signifie que la suite des arguments ne sont pas des options. Ceci est très pratique dans le cas ou l'un des

paramètres commence par un tiret, car autrement, le paramètre serait probablement traité comme une option invalide (la situation est encore pire si le paramètre porte le nom d'une option). Par exemple, si vous avez un fichier qui porte le nom `--help` et que vous voulez l'effacer, vous avez un problème, car c'est une option reconnue par la commande `rm` (qui affiche un message d'aide). Pour détruire le fichier, il faudra alors employer l'une des commandes suivantes :

```
$ rm -- --help
$ rm ./--help
```

Il peut arriver que certains arguments d'une commande ont des caractères spéciaux, comme le signe de dollars (qui sert à retourner la valeur d'une variable) ou un espace. Par exemple, si vous voulez afficher un signe de dollar suivi d'un `t` (sans espace), la commande suivante vous indiquera une erreur :

```
$ echo $t
t: Undefined variable.
```

Pire, si la variable `t` existe, la commande ci-dessus va afficher son contenu et non le `$t` désiré. Il est possible d'indiquer au *shell* qu'un caractère ne doit pas être interprété pour sa fonction en le préfixant du caractère \ (ce que l'on appelle en anglais le *escape character*). Ainsi, pour avoir le résultat désiré, il faudrait plutôt faire la commande suivante :

```
$ echo \$t
\$t
```

De même, en supposant que vous désirez renommer un fichier `toto` pour `to to`, vous feriez la commande suivante :

```
$ mv toto to\ to
```

car autrement, le *shell* interpréterait la ligne comme la commande `mv` avec trois arguments (`toto` et deux fois `to`). L'espace est le séparateur d'arguments. Il est également possible d'indiquer au *shell* de ne pas interpréter un suite de caractères ou de regrouper un ensemble de caractères séparés par des espaces. Les deux commande suivantes font respectivement exactement la même chose que les deux commandes précédentes :

```
$ echo '$t'

$ mv toto "to to"
```

Dans le premier cas, tout ce qui se trouve dans une paire d'apostrophes indique au *shell* de ne pas l'interpréter, i.e. de le prendre tel quel. Dans le cas de la paire de guillemets, ceci indique au *shell* de considérer ce qui s'y trouve entre comme un tout (i.e. un seul argument dans ce cas-ci). Dans le second exemple ci-haut, nous aurions pu aussi employer l'apostrophe.

Enfin, si jamais une commande est trop longue à votre goût, il est possible de la briser sur plus d'une ligne en terminant chaque ligne intermédiaire par le caractère \. Par exemple la commande suivante :

```
$ echo Voici une longue commande a briser
```

est équivalente à la commande brisée suivante :

```
$ echo Voici une longue\
> commande a briser
```

Pour effectivement briser la ligne, après le caractère \, il faut appuyer sur `[Enter]`, et ensuite le *shell* affiche un signe *« plus grand que »*, indiquant qu'il attend la suite de la commande.

Pour en savoir plus sur les *shells*, consulter leurs pages de manuel (`man bash` et `man tcsh`).

Information

Personnalisation de bash

Il est possible de personnaliser le comportement de votre interpréteur de commandes par l'intermédiaire d'un fichier d'initialisation qui se trouve dans le répertoire racine de votre compte : `.bashrc`

À chaque nouvelle instance d'un *shell*, le contenu du fichier est lu[1]. Ce fichier contient tout simplement une séquence de commandes reconnues par le *shell*, à raison d'une commande par ligne.

Il existe un autre fichier, `.profile`, qui est lu seulement pour le shell de démarrage d'une session. C'est un bon endroit pour y initialiser des variables. Un nouveau compte au DIRO possède un `.profile` comportant la commande :

```
. /usr/local/lib/Profile
```

Cette commande effectue les initialisations de base pour le DIRO. Un exemple de fichier `.profile` se trouve dans le répertoire en ligne de fichiers de SemUnix (voir à la page 3).

Personnalisation de tcsh

Dans le cas de `tcsh`, le fichier d'initialisation à utiliser est `.cshrc`.

Lorsque votre compte est initialement créé, le fichier `.cshrc` contient ceci :

```
source /usr/local/lib/Cshrc
```

[1]Sauf dans le cas où le *shell* n'est pas interactif, par exemple pour exécuter un script

Cette ligne exécute les commandes du fichier `/usr/local/lib/Cshrc`. Ce fichier contient les initialisations de bases pour le DIRO. Vous devez laisser cette première ligne telle quelle en tout temps.

Typiquement, les initialisations que l'on retrouve vont jouer sur les variables d'environnement (voir 6.2.2), les alias et certaines variables du *shell*. Vous configurez votre environnement pour qu'il soit adapté à votre façon de travailler. Un exemple de fichier `.cshrc` se trouve dans le répertoire en ligne de fichiers de SemUnix (voir à la page 3).

Lorsque vous apportez des changements à ce fichier, vous pouvez (et devriez) les mettre en action avec la commande suivante :

```
$ source ~/.cshrc
```

S'il y a des erreurs, la première vous sera signalée. Corrigez-la ! Alternativement, d'autres préfèrent fermer la fenêtre de terminal courante et en ouvrir une nouvelle.

Attention

Soyez prudent avec les changements que vous apportez à votre `.cshrc`. Le comportement de votre *shell* pourrait être étrange la prochaine fois que vous vous branchez (si vous n'avez pas fait le `source` prescrit et que vous n'avez pas corrigé les erreurs ou testé le nouvel environnement). Une précaution à prendre avant de modifier ce fichier est d'en faire une copie de sauvegarde. Si jamais, après avoir fait des changements dans votre fichier `.cshrc`, vous n'êtes plus capable de vous connecter sur une machine, choisissez **failsafe** comme type de session. Vous devriez alors être capable de vous connecter et de corriger vos erreurs.

Contrôle de tâches

Les *shells* permettent d'avoir un certain contrôle sur les tâches qui sont exécutées par le système. Entre autres, vous pouvez décider si une tâche doit s'exécuter en avant-plan ou en arrière-plan, ou même être suspendue. Ceci est très pratique dans un environnement où il n'y a pas de fenêtres (i.e. un environnement qui n'est pas graphique comme le système X Window). Dans un système avec fenêtres, il est assez facile d'ouvrir plusieurs fenêtres où chacune a une tâche associée. Dans un environnement qui n'a pas ces facilités (comme une connexion par modem, où les fenêtres ne sont pas disponibles), le contrôle sur les tâches qui est offert par le *shell* devient essentiel.

L'exécution d'une tâche en arrière-plan se fait en ajoutant le caractère & à la fin de la commande. À ce moment là, le *shell* répond en indiquant le numéro de la tâche et l'identificateur qui lui est associé. Par exemple :

```
$ arc &
[1] 1234
```

indique que la tâche `arc` est exécutée en arrière-plan. Le `[1]` indique que c'est la première tâche exécutée en arrière-plan et qu'elle est associée à l'identificateur 1234 (l'identificateur de processus, ou *process id*).

Si une tâche est exécutée en avant-plan, et que pour une raison quelconque nous voulons la mettre en arrière-plan, nous pouvons le faire en arrêtant la tâche courante habituellement en appuyant sur `[Ctrl-z]`. Le *shell* retourne un message disant *Suspended*, indiquant ainsi que la tâche est momentanément suspendue et reprendra au moment où l'usager donnera la commande de continuation d'exécution. L'usager peut faire cela de deux façons différentes.

La commande `bg` prend une tâche qui est suspendue et la met en arrière-plan. Si plusieurs tâches sont suspendues, nous pouvons spécifier laquelle doit être mise en arrière-plan en spécifiant le numéro de la tâche, précédé par le caractère `%`. Donc, la commande :

```
$ bg %2
```

prend la deuxième tâche suspendue et continue l'exécution en arrière-plan. Si aucun numéro de tâche n'est spécifié, la dernière tâche suspendue est mise en arrière-plan. Une syntaxe alternative est (pour l'exemple ci-avant) :

```
$ %2 &
```

i.e. prendre l'argument de `bg`, le mettre en premier et ajouter le `&` à la fin.

La commande `fg` est exécutée de la même façon que la commande `bg`, à la différence que la tâche spécifiée est maintenant exécutée en avant-plan et non en arrière-plan. Une syntaxe alternative est tout simplement comme la syntaxe alternative de `bg`, mais sans le `&` à la fin. Lorsqu'un commande est en avant-plan, il est souvent possible d'interrompre son exécution en appuyant sur `[Ctrl-c]`.

Il important de savoir qu'une tâche exécutée en arrière-plan cesse de fonctionner aussitôt qu'elle doit recevoir une entrée de `stdin` (voir 6.2.3). Dans un tel cas, l'entrée doit procéder par redirection. Si cela n'est pas désirable, il y a toujours moyen de ramener la tâche en avant-plan au moment où il faut lui fournir les données (à l'aide de la commande `fg`). Le *shell* informe l'usager du moment où un programme attend des données à partir de `stdin`, en indiquant la suspension d'exécution d'une tâche.

Enfin, voici d'autres commandes pratiques pour la manipulation des tâches :

`jobs` Cette commande affiche toutes les tâches exécutées en arrière-plan. Pour chaque tâche, la commande affiche son numéro, son statut (suspendue, terminée ou en exécution), et la commande de la tâche. Si l'option `-l` est donnée à la commande `jobs`, l'identificateur de processus est également affiché.

`kill` Tue les tâches (force la fin de leur exécution) données en arguments. La syntaxe des arguments est la même que pour les commandes `fg` et `bg`, mais en plus, il est possible de donner l'identificateur de processus à la place. Il peut arriver qu'une tâche s'obstine à ne pas vouloir être tuée. Dans ces cas-là, ajouter l'option `-9` à la commande. Ceci tue sans merci !

`stop` Met les tâches données en arguments en suspension. Les tâches sont spécifiées comme pour la commande `kill`.

Il existe une commande externe reliée aux tâches (plutôt aux processus, en fait) : `ps`. Invoquée sans argument, elle affiche des informations sur tous les processus contrôlés par le *shell* (les plus pertinentes étant l'identificateur de processus et la commande du processus). Avec l'option `-u $USER` (voir 6.2.2 pour la variable `USER`), elle affiche ces informations pour tous les processus de l'usager

sur la machine. Cette commande est souvent pratique entre autres pour tuer un processus qui n'a pas été démarré à partir d'un *shell* ou si le *shell* de départ a été perdu.

Information

Pour en savoir plus sur `ps`, consultez la *man page* de la commande (`man ps`).

6.2.2 Les variables d'environnement

Les variables d'environnement sont un moyen très utilisé de communiquer entre processus où de sauvegarder des préférences en tous genre. Le concept est le même que celui de variable dans tout langage de programmation : une variable d'environnement est simplement une *« case mémoire »* dans laquelle on peut ranger n'importe quelle chaîne de caractères. Votre *shell* vous permet de définir des variables d'environnement et d'en lire et changer la valeur.

La commande `set` permet d'afficher les variables d'environnement définies. On modifier une variables d'environnement simplement en lui affectant une valeur, mais cette valeur ne sear en vigeur que lors de la commande en cours. On peut aussi définir un variable pour toutes les futures commandes en utilisant la commande export. Définir (ou redéfinir) une variable dans un shell n'affecte pas la valeur que cette variable as dans un autre shell.

```
$ export LC_ALL=de_DE
$ rm a.abc
rm: Entfernen von a.abc nicht m\"oglich: Datei oder Verzeichnis nicht gefunden
$ LC_ALL=es_ES rm a.abc
rm: no se puede borrar "a.abc": No existe el fichero o el directorio
$ bash
$ export LC_ALL=en_EN
$ rm a.abc
rm: cannot remove `a.abc': No such file or directory
$ exit
$ rm a.abc
rm: Entfernen von "a.abc" nicht m\"oglich: Datei oder Verzeichnis nicht gefunden
$ export LC_ALL=fr_CA
$ rm a.abc
rm: ne peut enlever 'a.abc': Aucun fichier ou r\'epertoire de ce type
```

6.2.3 Flots d'entrée/sortie

Comme il a été dit en introduction de ce manuel, une des grandes forces du travail à la ligne de commande est qu'il permet de combiner des programmes d'une manière autrement impossible. Il a aussi été dit que les programmes UNIX sont souvent de petits utilitaires qui sont spécialisés dans une tâche particulière. Cette approche prend tout son sens quand l'usager peut combiner toutes ces petites applications de la manière qui lui convient.

Tout programme UNIX a au moins trois flots d'entrée/sortie qui lui permettent de communiquer avec l'usager et les autres programmes. On les appelle *Standard In*, *Standard Out* et *Standard Error*.

Les programmes peuvent ouvrir d'autres flots d'entrée/sortie mais ces trois-là sont toujours présents. Quand un programme veut imprimer quelque chose à l'écran, il envoie le texte à imprimer via son *Standard Out*. [2].

Les relais (*pipes*)

Prenons un programme simple que nous connaissons : `ls` (section 3.5.4). Ce programme lit le contenu d'un répertoire et l'imprime ; ainsi, quand `ls` vous affiche la liste de votre répertoire de travail, ce que vous voyez est le texte que `ls` a envoyé sur son *Standard Out*.

Prenons un autre programme que nous connaissons, `less` (section 4.6.1). Ce programme peut lire un fichier mais peut aussi lire ce qu'il reçoit via *Standard In*.

Supposons que nous voulions combiner les deux programmes. Par exemple, imaginons-nous sommes dans un répertoire qui contient un très grand nombre de fichiers, de sorte que l'appel à `ls` imprime des pages et des pages de texte, ce qui rend la lecture compliquée. Nous voudrions pouvoir tirer avantage des capacités de lecture page-par-page de `less`. La manière de combiner les commandes est assez simple :

```
$ ls -l | less
```

Le caractère de barre verticale « — » est appelé *pipe* (mot anglais signifiant « *tuyau* »). Son effet est de connecter les deux programmes. Quand le programme de gauche (`ls`, dans ce cas) enverra du texte dans son *Standard Out* pour l'imprimer, au lieu que le *shell* le reçoive et l'imprime à l'écran, c'est le programme de droite (`less`, dans ce cas) qui le recevra sur son *Standard In*.

Il est aussi possible de connecter plus d'un programme ensemble. Prenons la commande `grep`, qui sera montrée plus loin ; pour l'instant il suffit de savoir que si on lui donne un seul mot en paramètre, `grep` n'imprime sur son *Standard Out* que les lignes venant de son *Standard In* qui contiennent le mot donné. Ainsi, si nous ne voulons voir dans `less` que la description des fichiers dont le nom contient « *toto* », nous ferions ceci :

```
$ ls -l | grep toto | less
```

Il se passe donc ceci :

1. `ls` imprime la description des fichiers dans le répertoire courant ;
2. ces description sont passées à grep, qui n'imprime que les lignes contenant « *toto* » ;
3. ces lignes sont passées à `less` qui les affiche page par page.

Redirection vers des fichiers

UNIX permet aussi de rediriger la sortie d'un programme dans un fichier plutôt que vers un autre programme, ainsi que de lire à partir d'un fichier plutôt qu'à partir d'un autre programme.

Pour rediriger la sortie d'un programme dans un fichier, on utilise le caractère > suivi du nom du fichier. Par exemple, cette commande placera dans le fichier `liste` ce que `ls` affichera :

[2] Le mécanisme pour ce faire dépend du langage de programmation utilisé et un cours de programmation dépasse largement le cadre de ce manuel

```
$ ls > liste
```

Pour qu'un programme lise dans un fichier son *Standard In*, on utilise le <. Par exemple, pour n'afficher que les lignes du fichier `liste` qui contiennent le mot *« toto »*, on utilise la commande suivante. [3]

```
$ grep toto < liste
```

La redirection peut être combinée avec les *pipes*. Pour afficher via `less` les lignes dans le fichier `liste` qui contiennent le mot *« toto »* :

```
$ grep toto < liste | less
```

6.2.4 Manipulation de fichiers

Voyons maintenant quelques utilitaires qui, combinés à d'autres programmes via les *pipes*, peuvent être très utiles. Tous les utilitaires présentés dans cette section peuvent être utilisés d'au moins deux façons : soit en leur passant directement sur la ligne de commande la liste du(des) fichier(s) sur lequel(lesquels) agir, soit en leur donnant du texte sur lequel agir via *Standard In* (donc via un *pipe* ou une redirection de fichier).

grep

`grep` est sans contredit un des utilitaires les plus pratiques à la ligne de commande. Il sert à rechercher un *pattern* dans un texte. `grep` utilise les expressions régulières, décrites dans la section 6.2.7.

```
$ grep [options] regexp [fichiers]*
```

« regexp » est l'expression régulière à chercher. Si aucun fichier n'est donné, `grep` lit sur son *Standard In*.

Les options les plus courantes sont :

- **-r** Descendre dans les sous-répertoire (par récursion). Quand cette option est donnée, si certains des fichiers donnés sont des répertoires, `grep` va effectuer sa recherche dans tous les fichiers de ce répertoire, et recommencer avec les sous-répertoires, etc. Cette option ne peut être donnée que si au moins un fichier est donné en argument.
- **-n** Affiche le numéro de ligne devant chaque ligne imprimée.
- **-i** Ignorer la casse (i.e. : minuscules = majuscules).

Voici quelques exemples d'appels à `grep`.

Imprime toute les lignes contenant le mot *« any »* dans le fichier `/usr/include/stdio.h` :

```
$ grep any /usr/include/stdio.h
version 2.1 of the License, or (at your option) any later version.
```

[3] La commande `grep` peut en fait prendre des noms de fichier sur la ligne de commande. L'exemple ne sert qu'à démontrer les possibilités.

La même commande, avec l'option **-i** (ignorer majuscules/minuscules) :

```
$ grep -i -n any /usr/include/stdio.h
8:    version 2.1 of the License, or (at your option) any later version.
11:    but WITHOUT ANY WARRANTY; without even the implied warranty of
```

Autre manière d'appeler `grep` (les donnés sont passées via STDIN).

```
$ cat /usr/include/stdio.h | grep any
version 2.1 of the License, or (at your option) any later version.
```

find

Donne une liste de tous les fichiers contenus dans le répertoire donné en paramètre et ses sous-répertoires. Permet de limiter la liste avec plusieurs critères dont le nom de fichier, le type de fichier ou la date de modification. Permet d'exécuter une commande sur chacun de ces fichiers. Si un répertoire n'est pas donné le répertoire courant est utilisé.

```
$ find [chemin] [expression]
```

Sans aucune option ni paramètre, `find` affiche donc simplement la liste des fichiers contenus dans l'arbre des sous-répertoires du répertoire courant. Parmi les options les plus utilisées, on retrouve :

-name PATRON Limite la liste de fichiers à ceux dont le nom correspond au patron. Ce patron n'est pas un expression régulière mais suit les mêmes règles que l'expansion du shell.

-exec COMMANDE Execute la COMMANDE spécifiée sur chaque fichier trouvé.

Par exemple, cette commande donne tous les fichiers dans l'arbre des sous-répertoires de `/usr/include` dont le nom contient *« ctype* »* :

```
$ find /usr/include -name ctype*
/usr/include/linux/ctype.h
/usr/include/ctype.h
```

sort

Trie les lignes d'un fichier ou de l'entrée standard.

```
$ sort [options...] [fichiers]
```

Les options les plus courantes :

-n Considérer les entrées à trier comme étant numériques. Ainsi *« 8 »* est plus petit que *« 10 »* même si alphabétiquement l'ordre est inversé.

-f Ignorer la casse (i.e. : minuscules = majuscules).

-r Renverser l'ordre de tri.

-u Éliminer les entrées doubles.

Voyons un exemple pratique. La commande suivante affiche tous les fichiers du répertoire courant, en ordre de taille :

```
$ du -cs * | sort -n
10      avances.aux
11      guide_rapide.tex
11      programmes.aux
12      politiques.tex
12      semunix.log
16      style.tex
21      emacs-refcard.tex
24      semunix.pdf
26      bases_info.tex
54      bases_unix.tex
92      programmes.tex
123     avances.tex
534     semunix.dvi
7836    img
```

wc

`wc` (*Word Count*) compte le nombre de lignes, mots ou caractères dans un texte. Par défaut, affiche en ordre, le nombre de lignes, puis le nombre de mots et enfin, le nombre de caractères.

```
$ wc [options...] [fichiers]
```

-c Afficher le nombre de caractères seulement.

-l Afficher le nombre de lignes seulement.

-L Afficher le nombre de caractères de la ligne la plus longue.

-w Afficher le nombre de mots.

```
$ wc /usr/include/stdio.h
   639     3091    21452 /usr/include/stdio.h
```

6.2.5 Contrôle de processus

ps

Affiche la liste des processus actifs ou en veille s'exécutant sur le système. Par défaut, affiche tous les processus contrôlés par des terminaux de l'utilisateur en cours.

```
$ ps [options...] [processus]
```

-f (full) Afficher toute l'information.

-a Afficher tous les processus sauf les processus qui ne sont pas associés à un terminal.

-x Afficher même les processus n'étant pas contrôlés par des terminaux.

-u ¡utilisateur¿ Afficher les processus de l'utilisateur.

Les informations en sortie sont les suivantes :

PID (*Process IDentification*) Est le numéro unique de la tâche.

TTY Terminal à partir duquel le processus fut démarré.

STAT État du processus :

D En veille non-interruptible (habituellement un processus d'entrée-sortie).

R (*runnable*) En exécution.

S (*sleeping*) En veille.

T (*traced*) En mode débogage ou à l'arrêt.

Z (zombie) En mode *« zombie »*.

Pour BSD, d'autres formats peuvent être affichés :

W N'a pas de pages résidentes.

< Processus à haute priorité.

N Processus à faible priorité.

PPID (*Parent Process ID*) Identification du processus qui a démarré le processus listé.

CMD (CoMmanDe) Le nom de l'exécutable (avec ses paramètres de lancement en mode *« full »*).

STIME (*Start TIME*) Le moment où la tâche démarra.

```
$ ps
  PID TTY          TIME CMD
  527 pts/0    00:00:11 acroread
  800 pts/0    00:00:00 gv
  801 pts/0    00:00:07 gs
  915 pts/0    00:00:00 ps
$ ps -f 800
UID        PID  PPID  C STIME TTY      STAT   TIME CMD
semunix    800 17445  0 09:54 pts/0    S      0:00 gv semunix.ps
```

kill

Envoie un signal à un processus. Si aucun type de signal n'est spécifié, envoie par défaut *« TERM »* qui demande gentiment l'arrêt du processus en question. Si cela ne fonctionne pas, utilisez *« KILL »* mieux connu sous la forme *« -9 »*.

```
$ kill [ -s signal | -p ] [ -a ] [ -- ] pid ...

$ kill -l [ signal ]
```

-s Type de signal à envoyer, peut être un nom ou un numéro.

-l Afficher la liste des signaux possibles.

6.2.6 X Window

X Window est le système de fenêtrage standard de UNIX. C'est sur ce système que se basent des environnements de *desktop* comme KDE. X Window ne définit d'aucune façon de quoi ont l'air les fenêtres ou les éléments d'affichage, mais plutôt des concepts abstraits comme celui d'une fenêtre ou du curseur de souris.

Le plus grand avantage du système X Window est son architecture client-serveur. Contrairement à la quasi-totalité des systèmes d'interface graphique, avec X Window le programme qui affiche de l'information n'a pas besoin d'être exécuté sur le même ordinateur que celui auquel l'écran est connecté. Cela veut dire qu'avec sa permission, vous pouvez faire apparaître des fenêtres sur le *desktop* de votre voisin. Quand votre voisin commencera à utiliser la fenêtre, les données qu'il fournit (mouvements de souris, clics, texte tapé au clavier) seront acheminés à votre machine, donnés au programme, qui répondra avec les changement visuels appropriés (bouton prend le look *« enfoncé »*, le texte tapé apparaît à l'écran, etc).

Une autre application de cette architecture client-serveur est qu'elle permet de faire plusieurs postes de travail avec un seul ordinateur. Par exemple, dans le fond de la bibliothèque de Maths-Info se trouve une petite salle avec quatre postes de travail. Ces postes de travail ne sont pas reliés à un ordinateur, mais simplement une petite boîte attachée derrière l'écran. Cette petite boîte se connecte à une machine via le protocole X. Tout le travail de processeur se fait sur la machine distante : le poste local ne fait qu'afficher ce qui lui est donné.

L'ordinateur auquel l'écran est attaché est appelé le **serveur**. L'ordinateur sur lequel le programme roule est appelé le **client**. Cela semble l'inverse de la relation habituelle, mais peut être rationalisé en se disant qu'un serveur est une machine qui offre un service particulier, et que le serveur X offre les ressources de son écran.

Nous ne verrons pas ici les détails du protocole X Window. Voyons simplement quelques applications pratiques. Supposons que vous êtes connecté à la machine **blc21** et que vous voulez ouvrir une fenêtre sur l'écran de votre copain qui est connecté sur **blc22**. Votre copain doit d'abord vous donner la permission d'ouvrir des fenêtres sur son écran avec cette commande :

```
$ xhost + blc21
```

S'il avait voulu laisser quiconque ouvrir des fenêtres sur son écran, votre copain aurait pu taper ceci :

```
$ xhost +
```

Maintenant ouvrez une console et donnez à la variable d'environnement `DISPLAY` la valeur `blc22 :0` (voir la section 6.2.2). *« blc22 »* est la machine sur laquelle il faut afficher les fenêtres qui s'ouvriront, et *« 0 »* désigne le premier écran (de toute manière, la machine n'en a qu'un). À partir de maintenant, tout ce qui est démarré dans cette console s'affichera sur l'écran de votre voisin. Un bon moyen de tester cela :

```
$ xeyes
```

6.2.7 Expressions Régulières (ER)

Les expressions régulières sont omniprésentes dans le monde de la manipulation de chaînes de caractères. Plusieurs langages de programmation et plusieurs commandes UNIX en font grand usage. Les expressions régulières sont détaillées en profondeur dans le cours **IFT2105**.

Une expression régulière est la description d'une chaîne de caractères. Dans sa forme la plus simple, elle est elle-même une simple chaîne de caractères. Par exemple, *« toto »* est une expression régulière décrivant la chaîne de caractères *« toto »*.

On dit qu'une expression régulière *« correspond »* (*matches* en anglais) ou non à une chaîne de caractères. Quand on effectue une recherche avec une expression régulière, on essaie de la faire correspondre aux chaînes parmi lesquelles on cherche.

Là où les expressions régulières prennent toute leur puissance, c'est avec les caractères spéciaux qui correspondent à plus d'un caractère. Voici les caractères spéciaux les plus courants :

`.` Le point correspond à n'importe quel caractère. Ainsi, l'expression régulière `tat.` correspond aux chaînes *« tata »*, *« tate »*, *« tatt »*, etc.

`*` L'étoile indique que le caractère qui précède est répété zéro ou plus de fois. Par exemple, l'ER `ab*c` correspond aux chaînes *« ac »*, *« abc »*, *« abbc »*, etc.

`?` Le point d'interrogation indique que le caractère qui précède apparaît zéro ou une fois. Donc l'ER `ab?c` correspond uniquement aux chaînes *« ac »* et *« abc »*.

`[ ]` Les parenthèses carrées représentent un ensemble de caractères qui peuvent être utilisés un à la fois. Par exemple *« [aeo] »* dit que le caractère que les parenthèses carrées remplace peut être soit a, soit e ou soit o. Plus concrètement, si on avait l'expression *« hd[acd] »*, elle correspondrait à *« hda »*, *« hdc »* et *« hdd »*. Il est également possible, au lieu d'énumérer une longue liste de caractères qui se suivent dans l'ordre alphabétique, d'utiliser la notation *« [a-l] »* où *« a »* est la première lettre énumérée et *« l »* est la dernière.

Sous UNIX l'outil qui utilise le plus les expressions régulières est `grep` (section 6.2.4). Le manuel de `grep` donne plus de détails sur les expressions régulières.

6.3 Comment travailler de chez soi

6.3.1 Préalables

Avant de pouvoir travailler de chez vous, vous devez disposer d'un moyen de transférer de l'information depuis l'université vers la maison et vice versa. Si vous disposez d'un accès Internet par

l'intermédiaire d'un fournisseur tiers (ligne téléphonique ou haute vitesse), un canal de communication est déjà présent. Il devient alors possible de vous brancher sur le réseau du DIRO pour travailler à distance, consulter votre courrier électronique ou transférer des fichiers. Sans accès Internet, grâce à un modem téléphonique, il est possible de se brancher sur le réseau de l'université par le biais de Umnet. Sans modem téléphonique ni accès Internet, seul le transfert par disquettes et par stockage usb demeure possible.

Quelques logiciels peuvent s'avérer d'une grande utilité lors du travail sous Windows. Les éditeurs VI et Emacs sont disponibles pour cette plate-forme, ce qui vous permettra de profiter de leur puissance même depuis chez vous.

URL

La page Web de GNU Emacs vous permettra d'obtenir une version pour Windows
`http ://www.gnu.org/software/emacs/emacs.html`

URL

XEmacs constitue une intéressante alternative à GNU Emacs, car il contient un plus grand nombre de packages, dont AuC-TEX très utile à la rédaction de documents LATEX.
`http ://www.xemacs.org`

URL

GhostScript/GSView vous permettront d'ouvrir et imprimer des fichiers PostScript. Certains professeurs fournissent les énoncés des travaux pratiques et les notes de cours dans ce format, il est donc bon de pouvoir le lire.
`http ://www.cs.wisc.edu/~ghost`

6.3.2 Se connecter via SSH

Par le biais de l'Internet, vous pouvez obtenir une fenêtre de terminal distante. Grâce à cette console, vous pouvez travailler sur votre compte comme si vous vous trouviez sur le campus universitaire, à quelques restrictions près. Il est possible d'y taper des commandes Linux, compiler et exécuter des applications et consulter son courriel. Les commandes tapées sont acheminées au serveur du DIRO et exécutées sur ce serveur et non pas sur votre propre machine. La fenêtre de terminal est analogue à une fenêtre de terminal ouverte sur un poste du campus. Malheureusement, les applications en mode graphique, telles que le navigateur Internet ou les applets Java ne s'afficheront pas en tant que fenêtres sur votre ordinateur (à moins d'activer le protocole X, voir section 6.3.3).

La machine à laquelle vous vous connecterez est frontal.iro.umontreal.ca (voir section 5.3.2).

Le protocole SSH (Secure SHell) offre un cryptage des informations transmises, ce qui maximise la confidentialité de vos données, incluant votre mot de passe. Sous Windows, un client SSH n'est pas livré en standard, mais il existe plusieurs logiciels permettant de remplir cette fonction. Notre logiciel favori se nomme PuTTY et se présente sous la forme d'un simple exécutable, sans programme d'installation.

URL

Page de PuTTY, un excellent logiciel de connexion SSH
`http ://www.chiark.greenend.org.uk/~sgtatham/putty/`

Après avoir téléchargé gratuitement PuTTY, vous pouvez immédiatement le démarrer en exécutant `putty.exe`, ce qui ouvrira une fenêtre de configuration semblable à la figure 6.1.

Puisque le réseau du Diro utilise l'encodage UTF-8, il faut aller dans le menu du gauche sur *Translation*, puis changer ISO-8859-... par UTF-8. Ensuite, sélectionnez *Session* dans le menu de gauche (retourne à l'écran initial).

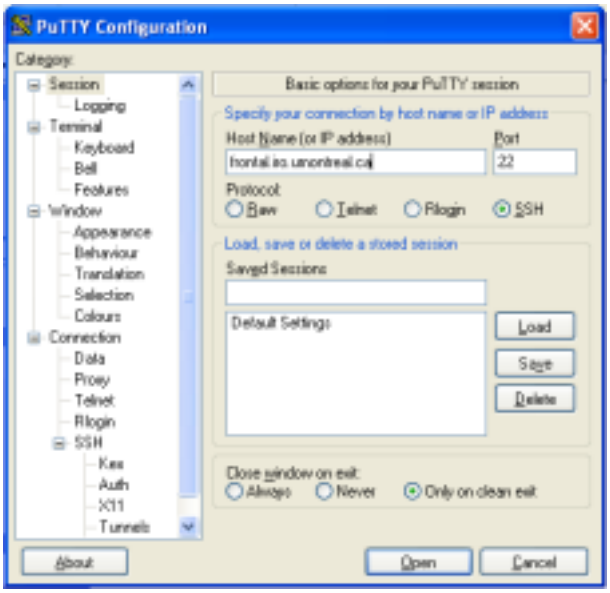


FIG. 6.1 – Fenêtre de branchement de PuTTY

Dans la liste de gauche de la fenêtre, sélectionnez **SSH** puis activez le bouton **2** pour indiquer à PuTTY d'utiliser la version 2 de SSH. Retournez à la section **Session** et tapez le nom du serveur dans le champ **Host Name**. Indiquez **SSH** comme protocole de connexion et cliquez sur le bouton **Open**. PuTTY offre également une foule d'autres options ainsi que la possibilité de sauvegarder les réglages effectués dans des profils pour un usage ultérieur. Au premier branchement, comme dans la figure 6.2, PuTTY vous indiquera que le serveur n'a pas pu être authentifié.

PuTTY Security Alert
The server's host key is not cached in the registry. You have no guarantee that the server is the computer you think it is.
The server's key fingerprint is:
1024 30:10:0f:18:4e:fa:6c:07:f8:13:75:be:f9:cc:fb:2d
If you trust this host, hit Yes to add the key to PuTTY's cache and carry on connecting.
If you want to carry on connecting just once, without adding the key to the cache, hit No.
If you do not trust this host, hit Cancel to abandon the connection.
Oui
Non
Annuler

FIG. 6.2 – Première connexion à un serveur via SSH

Cliquez sur **Oui** pour mémoriser la signature cryptographique de ce serveur, ce message ne sera alors plus jamais affiché lors de connexions subséquentes au DIRO. PuTTY affichera alors une fenêtre de texte dans laquelle votre nom d'utilisateur et votre mot de passe vous seront demandés. Lorsque

l'authentification est effectuée, vous disposez d'un écran de terminal qui constitue une voie d'accès à votre compte DIRO.

URL

Il existe d'autres logiciels permettant d'effectuer la connexion SSH. Ces logiciels ne sont pas tous égaux et gratuits.
`http ://www.openssh.org/windows.html`

6.3.3 Accès à l'interface graphique

En général, lorsqu'il est nécessaire d'accéder à une interface graphique, il vaut mieux travailler en local en raison de la grande quantité d'informations à transférer sur l'Internet. Par exemple, si vous travaillez sur une applet Java, il vaut mieux installer JDK chez vous et transférer les fichiers (voir section 6.3.4) pour compiler et exécuter l'applet sur votre propre système. Il arrive toutefois qu'il soit nécessaire d'accéder à une application Linux qui ne soit pas disponible sur votre système et dans ce cas, il est possible d'afficher l'interface graphique en utilisant la transparence réseau du X Window System.

Attention

Attendez-vous toutefois à un temps de réponse plutôt long, même avec une connexion Internet haute vitesse.

Pour afficher l'interface graphique, votre système doit avant tout être doté d'un serveur X.

Un serveur X constitue un logiciel interprétant un jeu de commandes standardisé et affichant une interface graphique. Par l'intermédiaire de la XLib, une ou plusieurs applications (clients) se connectent au serveur et envoient des requêtes d'affichage et de gestion des fenêtres à ce dernier. Le protocole X est multi-plateforme, ce qui permet de faire tourner un serveur X sur une machine et un client X sur une autre, peu importe si le système d'exploitation et même le processeur des deux machines est différent.

URL

Sous Windows, l'interface Graphic Device Interface (GDI) est utilisée plutôt que le protocole X Window System. C'est pourquoi un serveur X n'est pas inclus en standard. WeirdX constitue une excellente solution écrite en Java.
`http ://www.jcraft.com/weirdx`

Avant d'utiliser WeirdX, vous aurez besoin d'une machine virtuelle Java. Internet Explorer est livrée avec une telle machine virtuelle, mais elle ne supporte que JDK 1.1 et ne permet d'utiliser WeirdX que sous forme d'applet.

URL

La machine virtuelle Java de référence constitue l'implantation officielle de Sun que vous pouvez télécharger gratuitement.
`http ://java.sun.com/j2se/downloads.html`

WeirdX se présente sous la forme d'un fichier ZIP ou d'une archive TAR compressée. Des logiciels tels que WinZip seront en mesure de la décompacter. Vous pouvez décompresser WeirdX dans un répertoire de votre choix.

WinZip, un programme de compression ZIP en mesure de décompresser des archives au format TAR et des fichiers ZIP
`http ://www.winzip.com`

URL

Lorsque WeirdX est décompressé, vous pouvez aller modifier les paramètres dans les fichier-config/props dans un éditeur de texte de votre choix. Un des paramètres utiles à modifier est `weirdx.windowmode`. Le mode `Rootless` permet d'afficher chaque fenêtre X dans une seule et même fenêtre Windows.

Il est possible de démarrer WeirdX en utilisant la commande `java com/jcraft/weirdx/WeirdX` depuis l'emplacement où WeirdX est installé. Vous pouvez également démarrer le serveur en tant qu'applet dans votre navigateur Internet préféré. Dans ce cas, vous devrez ouvrir le fichier `misc/weirdx.html` dans le répertoire où est installé WeirdX depuis votre navigateur. Il sera nécessaire d'éditer ce fichier afin que les paramètres de l'applet reflètent ceux trouvés dans `config/props`.

Après le démarrage de WeirdX, une fenêtre vide s'affichera, le serveur X est alors prêt ! Le serveur tourne par défaut sur l'écran 0 de l'affichage 2. Un port de communication TCP a également été ouvert afin de recevoir les requêtes extérieures. Votre serveur X n'accepte pas des connexions de n'importe quelle machine. Ainsi, vous n'aurez pas à craindre qu'un utilisateur mal intentionné n'éprouve l'envie de faire ouvrir une centaine de fenêtre afin de vous bombarder d'informations et faire figer votre système. La liste des serveurs autorisés est donné par le paramètre weirdx.display.acl.

Dans votre client SSH aller activer le *X11 forwarding* et mettez **localhost :2.0** comme serveur X. Dans PuTTY, allez dans **SSH** et ensuite **X11**, avant d'établir une connexion (voir figure 6.3).

Établissez ensuite une connexion SSH sur le serveur frontal. Lorsque le réglage est effectué, toute application à interface graphique que vous lancerez depuis votre serveur attitré au DIRO verra son contenu apparaître dans la fenêtre de WeirdX. Le serveur frontal n'a pas besoin d'être dans le weirdx.display.acl puisque SSH fait un tunnel entre le serveur frontal et votre machine. WeirdX voit donc les requêtes comme parvenant de votre propre machine.

Il ne faut pas confondre les deux types de serveurs mis en jeu lors de cette connexion. En temps normal, frontal constitue votre serveur et votre machine s'identifie comme un client. Dans le cas de X, votre machine devient un serveur et une application distante constitue un client (voir section 6.2.6).

Information

Sous Linux, un serveur X très complet du nom de XFree86 (ou maintenant Xorg) est livré en standard et peut être utilisé pour afficher des fenêtres distantes. Pour y parvenir, établissez simplement une connexion SSH depuis une fenêtre XTerm ou Konsole. vous pourrez alors démarrer des applications à interface graphique à distance. Si cela ne fonctionne pas, vous pouvez tenter une connexion un peu plus manuelle. Tout d'abord, vous devrez autoriser votre serveur X à afficher des fenêtres sur votre machine. La commande `xhost +frontal.iro.umontreal.ca` autorisera l'accès depuis frontal. Ensuite, vous devrez établir une connexion SSH et, sur le serveur du DIRO, définir

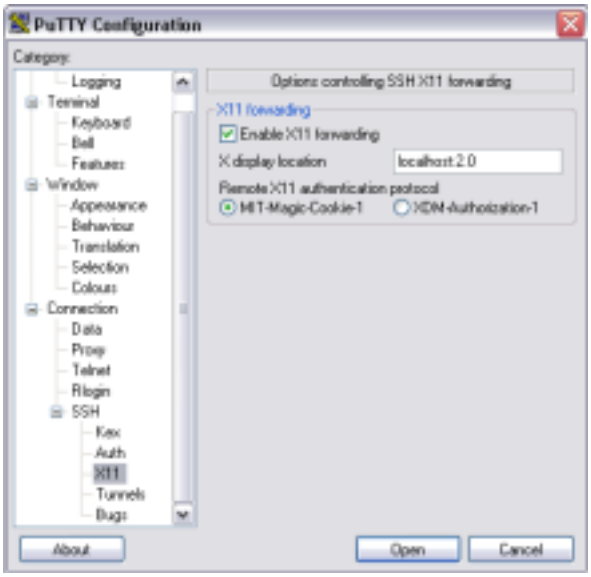


FIG. 6.3 – Activer le "X11 forwarding" et spécifier "localhost :2.0" pour "X display location" dans PuTTY

la variable `DISPLAY` afin qu'elle pointe vers <ip> `:0.0`. En Pour obtenir un écran de branchement, vous pouvez utiliser la commande `X -query` <serveur> `:1.0`. Ce qui démarrera un serveur X sur l'affichage 1.

Une méthode alternative serait d'utiliser le *X11 forwarding*. Ajouter l'option -X lorsque vous établissez une connexion avec le client ssh. La commande `ssh -X frontal.iro.umontreal.ca` fera l'affaire.

Information

Le *X11 forwarding* as l'avantage de traverser n'importe quelle passerelle et qu'il ne nécessite pas de définir de permissions pour la machine distante puisque les connexions semble venir de votre propre machine

Information

Le protocole SSH permet de rediriger d'autres port TCP comme le port 80 (http). Pour plus d'informations, consulter `man ssh` pour les arguments `-L` et `-R`.

6.3.4 Comment transférer des fichiers

Si vous souhaitez travailler chez vous et que vous avez besoin de transférer des fichiers vers le DIRO, deux solutions s'offrent à vous : sftp ou scp. Depuis peu, l'administration encourage fortement d'utiliser des connexions sécurisées (encryptées) pour vous protéger d'attaques telles que le vol de mot de passe, qui pourrait ensuite servir à compromettre la sécurité du système informatique du département au complet !

Nous avons déjà expliqué l'utilisation de `scp` dans Linux à la section 4.1.3. Si vous travaillez dans Windows à la maison, il vous est possible d'utiliser ces méthodes de transfert. Plusieurs clients existent pour effectuer les transferts. Nous vous conseillons un programme gratuit, distribué sous la license GPL : WinSCP.

Page web de WinSCP, si vous voulez vous le procurer
`http ://winscp.net/eng/index.php`

URL

Dans la fenêtre de transferts de WinSCP, vous avez deux sections. Celle de gauche représente les contenus du premier ordinateur. Celui-ci peut être votre ordinateur (connexion locale) ou un ordinateur distant. La section de droite est la section du deuxième ordinateur qui peut être également une connexion locale ou un hôte distant. Pour vous connecter localement, tapez localhost dans une des deux barres d'adresse. Pour vous connecter à distance, tapez `utilisateur@serveur.domaine` où `utilisateur` est votre nom d'utilisateur sur la machine ; par exemple votre *login* du DIROsi vous vous connectez au département. Quand à `serveur.domaine`, c'est le nom de la machine que vous voulez rejoindre. Vous pouvez entrer une adresse DNS ou une adresse IP correspondant à la machine en question. (Par exemple, pour vous brancher au DIRO, vous devez entrer frontal.iro.umontreal.ca)

Pour transférer des fichiers, une fois que vous êtes connecté aux deux hôtes, vous n'avez quà glisser une sélection de fichiers d'un côté à l'autre.

Si vous souhaitez travailler chez vous sans établir une connexion SSH, vous devrez, à un certain moment, transférer votre travail sur votre compte DIRO afin de le remettre. Ce transfert s'effectue alors par FTP. Sous Windows, aucun client FTP n'est installé par défaut (voir la section 4.1.3).

6.3.5 Comment consulter son courrier électronique

Il existe plusieurs façons de consulter son courriel électronique chez soi. La technique la plus simple consiste à établir une connexion SSH vers le DIRO et d'utiliser Pine. Malheureusement, il est difficile de gérer les pièces jointes, car elles doivent être transférées manuellement par FTP vers votre ordinateur domestique pour être consultées. Les messages HTML perdent de leur côté beaucoup de leur richesse. De plus, la connexion SSH est parfois lente et il devient peu convivial de consulter et envoyer du courrier par ce moyen.

Le protocole IMAP permet quant à lui d'utiliser la puissance de votre machine domestique de façon plus optimale en partageant davantage le travail entre le serveur au DIRO et votre système. IMAP s'apparente à POP3, le protocole utilisé par la plupart des fournisseurs d'accès Internet.

Pour utiliser IMAP, vous devrez disposer d'un logiciel de courrier électronique. Sous Windows, le plus courant s'avère sans doute Microsoft Outlook Express. Ce dernier est livré avec Internet Explorer et, à partir de Windows 98, est livré en standard. Il est également possible d'utiliser Mozilla, Thunderbird ou tout autre logiciel de courriel. Sous Linux, Mozilla et Thunderbird sont aussi utilisables. Sylpheed constitue également un très bon choix de logiciel de courrier électronique.

Mozilla, navigateur Internet et logiciel de courrier électronique pour Linux et Windows
`http ://www.mozilla.org`

URL

Thunderbird, logiciel de courrier électronique pour Linux et Windows
`http ://www.mozilla.org/products/thunderbird/`

URL

URL

Sylpheed, logiciel de courrier électronique pour Linux
`http ://sylpheed.good-day.net`

La configuration du courrier électronique dépend du logiciel utilisé. Dans tous les cas, des informations vitales seront nécessaires.

SMTP — Nom du serveur utilisé pour envoyer du courrier électronique. Si vous utilisez `smtp.iro.umontreal.ca`, vous ne pourrez envoyer du courrier que vers les usagers du DIRO, car toute adresse ne se terminant pas par `@iro.umontreal.ca` sera rejetée d'office. Ici, vous pouvez utiliser le serveur SMTP de votre fournisseur Internet ou `magellan.umontreal.ca`.

IMAP — C'est le serveur de courrier entrant, votre boîte aux lettres virtuelle. Utilisez ici `imap.iro.umontreal.ca`.

Utilisateur — Le nom d'utilisateur à fournir à votre client de courrier électronique constitue votre nom d'usager DIRO qui sera passé au serveur IMAP pour identifier la boîte aux lettres à consulter.

Mot de passe — Il est recommandé de laisser ce champ vide afin que votre logiciel de courrier électronique demande votre mot de passe lors de chaque branchement. Vous pouvez également spécifier votre mot de passe DIRO afin d'automatiser la connexion, mais cela ouvrira une brèche de sécurité, car ce mot de passe sera stocké d'une façon ou d'une autre sur votre disque dur. Bien qu'en théorie il soit impossible à obtenir, une personne mal intentionnée pourrait faire différer la pratique de la théorie.

Si votre logiciel de courrier électronique vous permet de modifier les ports SMTP et IMAP, ne le faites pas dans le cas des serveurs de l'Université de Montréal. Le serveur du DIRO ainsi que Magellan utilisent les ports standards. Vous pouvez ou non activer SSL pour SMTP et IMAP, toutes les autres options SMTP et IMAP devraient demeurer inchangées.

En guise d'exemple, nous allons exposer la procédure de configuration du logiciel Outlook Express. Tout d'abord, démarrez le logiciel. Au premier démarrage, il devrait vous demander de configurer votre courrier électronique. Si tel n'est pas le cas, utilisez le menu **Outils** pour accéder à l'option **Comptes**. Dans la boîte de dialogue apparaissant à l'écran, sélectionnez l'onglet **Courrier** puis cliquez sur **Ajouter**, puis **Courrier...**. Les figures 6.4, 6.5, 6.6 et 6.7 montrent les étapes importantes imposée par l'assistant de configuration d'un compte de courrier électronique.

Lorsque la configuration est terminée, activez le menu **Outils**, options **Comptes**, si ce n'est déjà fait, et double-cliquez sur le compte nouvellement créé. Les figures 6.8 et 6.9 montrent les onglets importants à modifier.

Fermez les propriétés du compte puis la boîte de dialogue des comptes. Le message *« Voulez-vous télécharger les dossiers à partir du serveur de messagerie que vous avez ajouté ? »* sera affiché, répondez **Oui** et les dossiers IMAP seront téléchargés comme dans la figure 6.10. Outlook Express est désormais prêt à être employé avec votre compte DIRO !

6.3.6 Cygwin, une couche UNIX au-dessus de Windows

Si vous ne souhaitez pas installer Linux, vous pouvez tout de même accéder à un sous-ensemble de ses fonctionnalités grâce au logiciel Cygwin. Le cœur de Cygwin est formé d'un fichier DLL four-

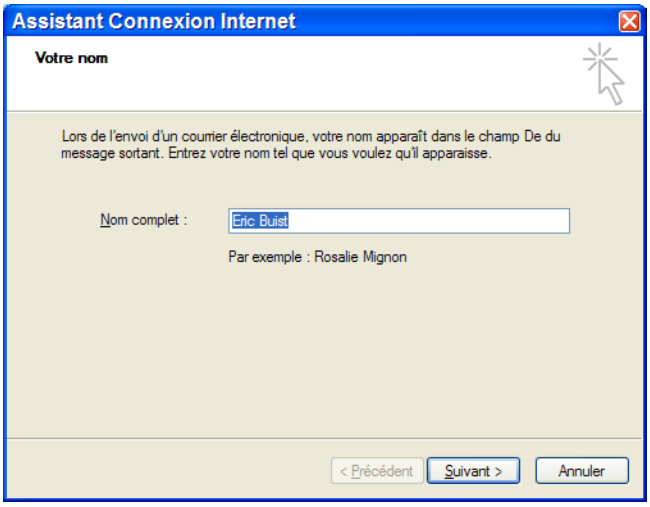

Assistant Connexion Internet
Votre nom
Lors de l'envoi d'un courrier électronique, votre nom apparaît dans le champ De du message sortant. Entrez votre nom tel que vous voulez qu'il apparaisse.
Nom complet :
Eric Buist
Par exemple : Rosalie Mignon
< Précédent
Suivant >
Annuler

FIG. 6.4 – Votre nom complet vous sera demandé.

nissant une implantation des fonctions POSIX au-dessus de Windows, ce qui permet la compilation et l'utilisation de plusieurs applications disponibles sous Linux. Entre autres, vous y trouverez le compilateur GCC, le *shell* Bash, les commandes `find`, `grep`, `tar` et d'autres encore. Cygwin permet d'utiliser GCC pour compiler la plupart de vos travaux pratiques en C/C++, et cela, de chez vous, sans aucune connexion SSH. Cygwin est également pourvu d'un client SSH (commande `ssh`).

Cygwin n'est pas Linux pour Windows ! En utilisant Cygwin, vous hériterez des bogues et instabilités inhérentes à Windows. Vous ne pourrez profiter de l'architecture pleinement 32 bits et de l'unique robustesse d'un véritable noyau Linux.

Attention

L'installation de Cygwin est très simple, il suffit de télécharger le programme d'installation qui, à son tour, téléchargera les fichiers requis. Vous pouvez aussi exiger de télécharger Cygwin intégralement et l'installer à volonté sur n'importe quel système Windows. Toutefois, vous éprouverez peut-être des problèmes de performance et de stabilité si vous tentez d'utiliser Cygwin sous Windows Me. Cygwin occupera une centaine de méga-octets sur votre disque dur.

Cygwin
`http ://www.cygwin.com`

URL

Il existe un petit problème lié à Cygwin, soit la saisie des accents. Pour le surmonter, depuis le *shell* de Cygwin, tapez `cd` puis `vi .inputrc`. Sous VI, tapez `[a]` puis les lignes suivantes.

```
set meta-flag on
```

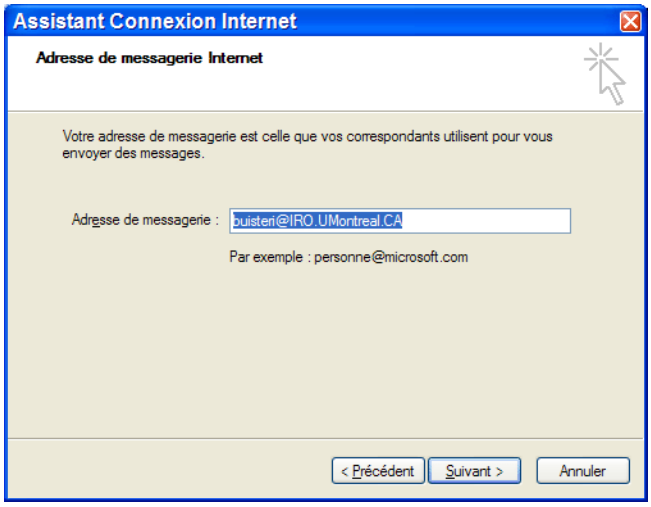

Assistant Connexion Internet
Adresse de messagerie Internet
Votre adresse de messagerie est celle que vos correspondants utilisent pour vous envoyer des messages.
Adresse de messagerie :
buisteri@IRO.UMontreal.CA
Par exemple : personne@microsoft.com
< Précédent
Suivant >
Annuler

FIG. 6.5 – Votre adresse de courrier électronique vous sera demandée ; spécifiez login@IRO.UMontreal.CA

```
set convert-meta off
set output-meta on
```

Appuyez sur `[Esc]` puis sur tapez `:wq`, suivi de `[Entrée]`. Tapez `exit` pour quitter Cygwin, redémarrez et tout devrait fonctionner.

Les nouvelles versions de Cygwin sont également livrées avec un portage de XFree86. Pour l'utiliser, tapez simplement `/usr/X11R6/bin/startxwin.bat`. Il peut être utilisé en lieu et place de WeirdX pour établir une connexion avec interface graphique (voir section 6.3.3).

6.3.7 Magellan

Cette section ne s'adresse qu'à ceux qui ne sont abonnés à aucun fournisseur Internet privé. Avec un tel fournisseur, il est plus aisé d'accéder au réseau avec un navigateur Web, un logiciel SSH et un logiciel FTP. Une connexion au parc de modems peut également dépanner en cas de panne de votre fournisseur à un moment critique.

Magellan constitue un service de la DGTIC de l'Université de Montréal et non un service du DIRO. Il offre la possibilité de vous brancher au réseau universitaire par modem et d'obtenir une adresse de courrier électronique.

URL

Site de la Direction générale des technologies de l'information et de la communication (DGTIC)
`http ://www.dgtic.umontreal.ca`

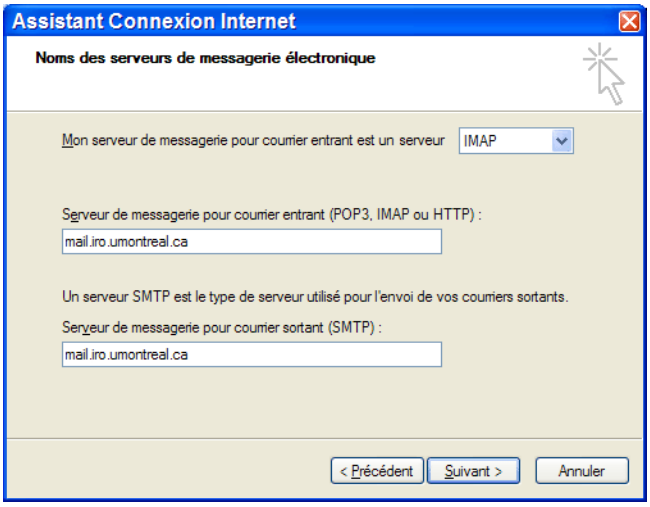

Assistant Connexion Internet
Noms des serveurs de messagerie électronique
Mon serveur de messagerie pour courrier entrant est un serveur
IMAP
Serveur de messagerie pour courrier entrant (POP3, IMAP ou HTTP) :
mail.iro.umontreal.ca
Un serveur SMTP est le type de serveur utilisé pour l'envoi de vos courriers sortants.
Serveur de messagerie pour courrier sortant (SMTP) :
mail.iro.umontreal.ca
< Précédent
Suivant >
Annuler

FIG. 6.6 – Indiquez que vous utilisez un serveur IMAP et entrez les noms des serveurs IMAP et SMTP.

Pour accéder à Magellan, vous devrez ouvrir un compte sur le système en utilisant votre profil informatique de la DGTIC. Il est possible d'accéder à ce profil en utilisant votre code permanent et votre UNIP. À partir du profil informatique, accessible par le biais du site Web de la DGTIC, vous pourrez obtenir un nom d'utilisateur et un mot de passe initial pour accéder au parc de modems et au courriel Magellan. Il sera ensuite nécessaire de changer ce mot de passe.

Le parc de modems, le courriel Magellan et le réseau DIRO constituent trois services distincts. Votre nom d'usager et votre mot de passe initial seront identiques pour tous les services de la DGTIC, mais ils différeront dans le cas du DIRO. Le changement d'un mot de passe pour un service n'influence pas les autres. Il est possible de modifier votre mot de passe de tous les services de la DGTIC simultanément par le biais du profil informatique de la DGTIC, mais le mot de passe DIRO est indépendant de la DGTIC.

Information

Avant d'effectuer le branchement, assurez-vous que votre modem est installé et configuré correctement sous le système d'exploitation à utiliser.

Selon votre localité, des frais d'interurbain pourraient être imposés lors du branchement par ligne téléphonique.

Attention

Branchement en mode PPP

Cette connexion ne permet pas d'accéder à Internet en mode graphique, exception faite du domaine `umontreal.ca`. Pour accéder à un tout autre site Internet, utilisez une connexion SSH sur

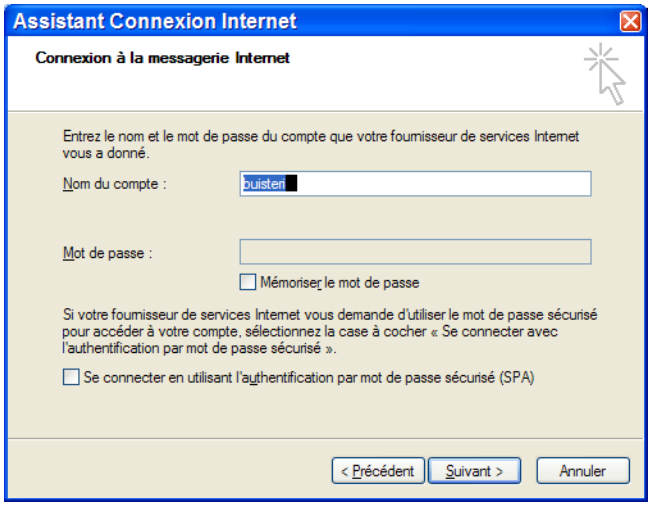

Assistant Connexion Internet
Connexion à la messagerie Internet
Entrez le nom et le mot de passe du compte que votre fournisseur de services Internet vous a donné.
Nom du compte :
Mot de passe :
Mémoriser le mot de passe
Si votre fournisseur de services Internet vous demande d'utiliser le mot de passe sécurisé pour accéder à votre compte, sélectionnez la case à cocher « Se connecter avec l'authentification par mot de passe sécurisé ».
Se connecter en utilisant l'authentification par mot de passe sécurisé (SPA)
< Précédent
Suivant >
Annuler

FIG. 6.7 – En guise de nom de compte, entrez votre nom d'usager et décochez la case Mémoriser le mot de passe.

frontal et le navigateur en mode texte Lynx. Toutefois, cet accès est plutôt lent et il est recommandé de s'abonner à un fournisseur privé pour un usage plus intensif de l'Internet.

La procédure de connexion dépend du système d'exploitation utilisé et exige plusieurs manipulations. Sous Windows XP, accédez au **Panneau de Configuration** et double-cliquez sur l'icône **Connexions réseau** (figure 6.11). Voyez plus loin si vous utilisez une autre version de Windows.

Cliquez sur **Créer une nouvelle connexion**, un assistant apparaîtra alors. Cliquez sur **Suivant** pour passer le premier écran explicatif puis sélectionnez **Établir une connexion à Internet**. À l'étape suivante, choisissez **Configurer ma connexion manuellement**. L'assistant vous demandera ensuite le nom du fournisseur Internet ; spécifier `Magellan`, mais ce nom n'importe pas. Le numéro de téléphone sera ensuite demandé ; indiquez `5143432411` (`*70,5143432411` si vous êtes abonnés au service d'appel en attente). Ne spécifiez pas de nom d'utilisateur ni de mot de passe. La configuration initiale étant effectuée, une icône **Magellan** apparaîtra dans les **Connexions réseau** et pourra être utilisée pour toute connexion ultérieure. La figure 6.12 présente la fenêtre affichée lors du double-clic sur la connexion.

- Lors du premier branchement, cliquez sur **Propriétés** afin d'affiner les réglages. Sous l'onglet **Sécurité**, cochez la case **Afficher une fenêtre de terminal** (voir figure 6.13).
- Sous l'onglet **Gestion de réseau**, assurez-vous que le type de serveur soit PPP (voir figure 6.14).
- Cliquez sur le bouton **Paramètres** près de PPP et décochez la case **Activer les extensions LCP** dans la fenêtre qui surgira et qui est semblable à la figure 6.15.
- De nouveau sous l'onglet **Gestion de réseau**, cliquez sur **Protocole Internet (TCP/IP)** puis sur le bouton **Propriétés** au bas de la liste des protocoles. Dans la fenêtre qui surgit, cochez **Obtenir une adresse IP automatiquement** et **Utiliser l'adresse de serveur DNS suivante**. Spécifiez `132.204.2.103` comme DNS primaire et `132.204.2.102` comme

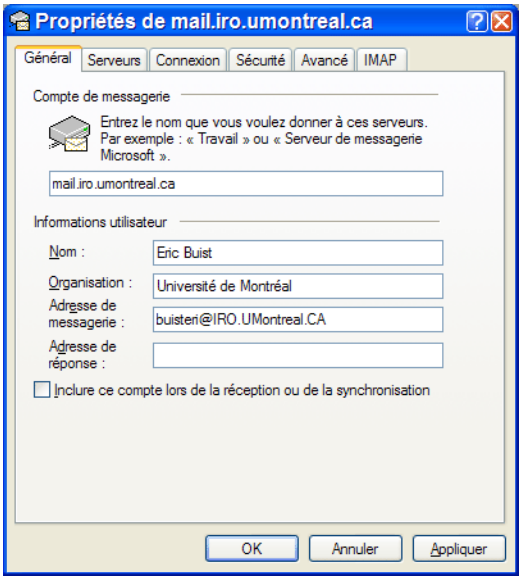


FIG. 6.8 – Sous l'onglet Général, vous pouvez définir votre organisation en tant que l'Université de Montréal.

DNS secondaire. (voir figure 6.16)

- Utilisez ensuite le bouton **Avancé** pour accéder à une nouvelle boîte de propriétés. Cochez-y la case **Passerelle par défaut** et décochez **Utiliser la compression d'en-têtes IP**. Cliquez ensuite sur **OK** jusqu'à revenir à la boîte de connexion à Magellan.
- Assurez-vous qu'aucun nom d'utilisateur ou mot de passe n'est spécifié et cliquez sur **Numéroter**. Le modem se mettra alors en marche et si tout va bien, une fenêtre de terminal apparaîtra. Tapez votre nom d'usager Magellan et votre mot de passe afin d'atteindre l'invite `umnet>` et tapez-y `ppp default`. Lorsque des caractères insensés se mettront à défiler, cliquez sur **Continuer** et la connexion devrait s'établir.

Sous Windows 2000, la procédure est très similaire tandis qu'elle diffère légèrement sous Windows 95/98/Me. Dans le second cas, vous devrez vous assurer que l'**Accès réseau à distance** est bien installé et y accéder en double-cliquant sur l'icône correspondante dans le **Poste de travail**. Les mêmes étapes de mise en œuvre s'appliquent, mais les noms des options et paramètres diffèrent légèrement.

La DGTIC constitue la source d'informations de référence au sujet du parc de modems
`http ://www.dgtic.umontreal.ca/html/pages/009.htm`

URL

Sous Linux, en supposant que PPP est installé dans votre noyau, ce qui est le cas avec la version RedHat 7.3, vous devrez tout d'abord devenir super-utilisateur (*root*) en utilisant la commande `su` ou en vous branchant sous le nom d'usager root. Éditez le fichier `/etc/resolv.conf` et ajoutez-y

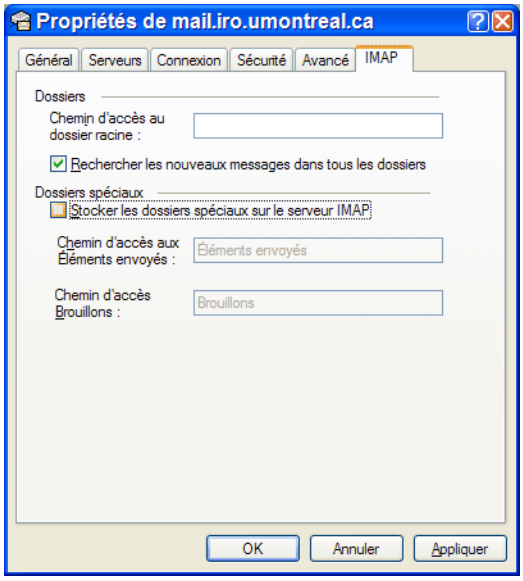


FIG. 6.9 – Sous l'onglet IMAP, décocher Stocker les dossiers spéciaux sur le serveur IMAP afin de stocker les messages envoyés et les brouillons sur votre machine. Si vous laissez cette option cochée, nous recommandons d'indiquer d'utiliser sent-mail et saved-messages plutôt qu'Élements envoyés et Brouillons.

les deux lignes suivantes

```
nameserver 132.204.2.103
nameserver 132.204.2.102
```

Démarrez ensuite `minicom -s` afin de configurer MiniCOM. Indiquez-lui l'emplacement de votre modem ainsi que ses paramètres. Considérons que le modem se trouve sur `/dev/modem`.

Lors de chaque branchement, exécutez la commande `minicom` pour faire apparaître le logiciel de communication. À l'intérieur de MiniCOM, tapez la commande `ATDT5143432411` ou `ATDT*70,5143432411` si vous êtes abonné à l'appel en attente, puis validez par `[Entrée]`. Le modem se mettra alors en fonction. Effectuez le branchement pour atteindre l'invite `umnet>` puis tapez `ppp default`. Utilisez la touche `[CTRL-A]` puis `[Q]` afin de quitter MiniCOM sans réinitialiser le modem. Finalement, tapez `/usr/sbin/pppd -d -detach /dev/modem`.

Lorsque la connexion est établie, vous pouvez redevenir un utilisateur normal. Pour terminer la connexion, redevenez root et invoquez `/etc/ppp/ppp-off`.

URL

Pour en savoir plus sur la configuration de PPP, consultez le PPP HOWTO
`http ://www.tldp.org/`

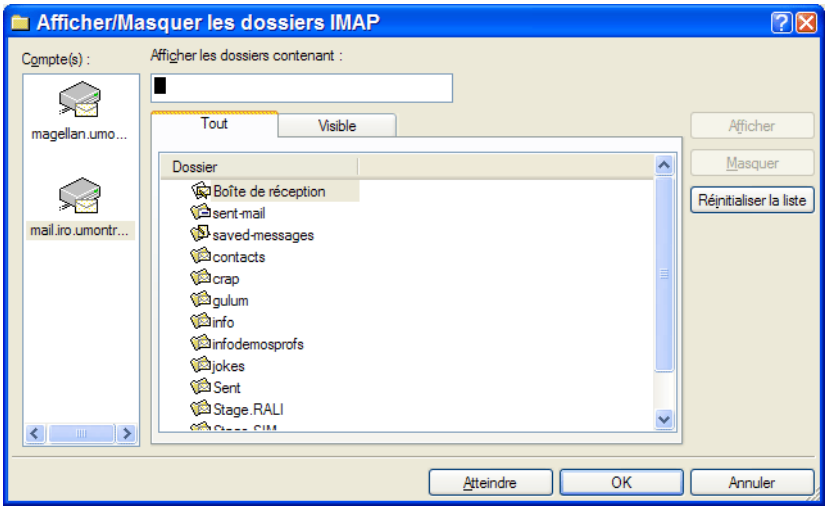


FIG. 6.10 – Dossiers IMAP téléchargés

6.4 Comment installer Linux chez soi

Dans certains cas, Linux s'avère incontournable. Par exemple, si vous devez écrire des applications Qt pour un travail pratique donné, sans Microsoft Visual C++ 6, vous ne pourrez pas utiliser la version non commerciale de Qt pour Windows. Sous Linux, Qt est gratuit pour usage non commercial et fonctionne de concert avec un compilateur libre, soit G++ (GCC pour le C++). L'installation de Linux procure également une expérience unique d'exploration de votre machine, ainsi qu'un système d'exploitation plus stable que Windows.

L'installation de Linux constitue une opération complexe qui exige temps et effort. Il est recommandé de ne pas l'effectuer en période de stress.

Attention

L'installation de Linux nécessite un certain nombre de préparatifs et de manipulations. Après avoir choisi une distribution, il faudra vous la procurer. Après avoir effectué une sauvegarde de vos données sensibles et une analyse du matériel de votre système, vous devrez réaménager votre disque dur puis effectuer l'installation proprement dite. Ensuite viendront un grand nombre d'ajustements et de réglages afin de surmonter les problèmes qui surgiront et rendre le système optimal.

6.4.1 Quelle distribution choisir ?

Linux est distribué de façon libre et décentralisée. Contrairement à des systèmes propriétaires tels que Microsoft Windows, plusieurs compagnies offrent des distributions Linux. On appelle dis-

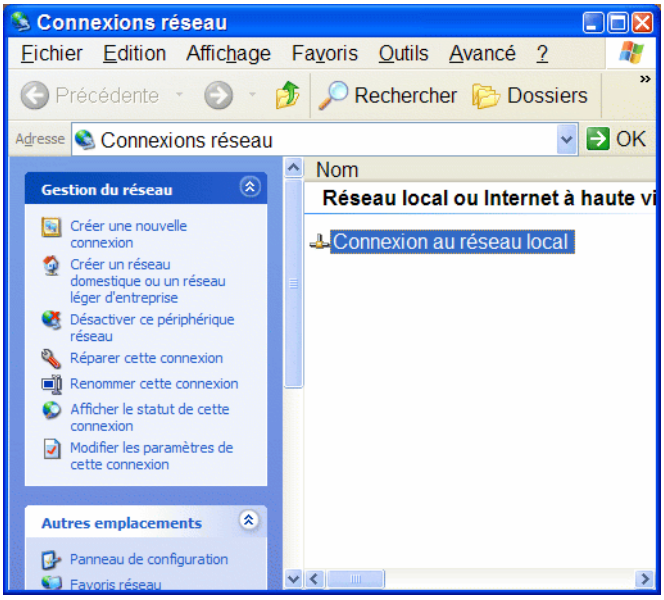


FIG. 6.11 – Connexions réseau de Windows XP

tribution Linux un produit comportant tout le nécessaire pour installer Linux chez soi, s'y initier et apprendre à l'utiliser ainsi que l'administrer de façon efficace. Bien que Linux soit un logiciel libre, il existe une concurrence quant aux distributions, certaines offrant des fonctionnalités que d'autres n'offrent pas. Sachez toutefois que le cœur du système (*kernel*) et son architecture demeurent les mêmes et tout logiciel compilé sur une distribution devrait théoriquement tourner sous une autre si toutes les librairies nécessaires sont installées. Les distributions diffèrent par les applications livrées, la convivialité de leurs interfaces, la gestion de l'installation et de la désinstallation ainsi que la documentation.

Il existe un grand nombre de distributions disponibles et en faire l'énumération exhaustive serait difficile. Nous nous contentons donc d'indiquer les principales avec leurs caractéristiques majeures.

Fedora Core (anciennement RedHat) La plus répandue et utilisée dans l'industrie et les universités. Elle offre une installation simple en mode graphique. Il existe un certain nombre d'outils de configuration, mais on ne trouve aucun panneau de contrôle les intégrant tous. Les outils d'installation et de configuration disponibles sont conçus à l'aide des librairies GTK+ et GNOME. Fedora Core est une version de RedHat supportée par la communauté, et non pas par RedHat lui même. Le DIROutilise une version récente de Fedora Core.

Mandriva (anciennement Mandrake) Mandrake offre un processus d'installation et de configuration simplifié et, pour un utilisateur non initié, constitue peut-être le choix idéal. Mandrake privilégie KDE comme environnement de travail et ses outils d'installation/configuration se basent sur ses librairies. Mandrake offre également une possibilité d'installer Linux sur une partition Windows, ce

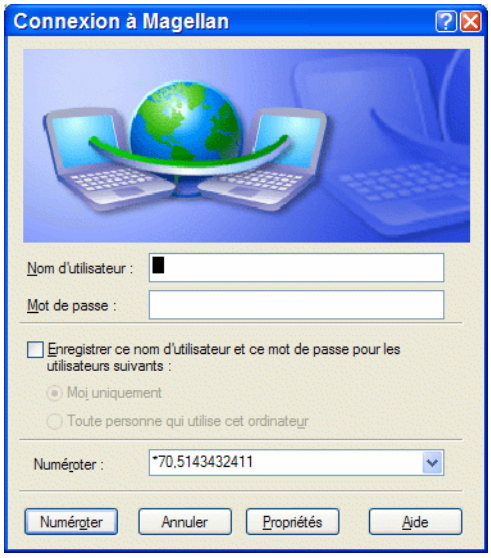


FIG. 6.12 – Écran de connexion à Magellan par PPP

qui évite le partitionnement, mais les performances de cette option laissent à désirer (lire, à proscrire). La compagnie Mandrake Soft a récemment fusionné avec Connectiva Linux pour former MandrivaSoft. Le nom de la distribution a également changé pour Mandriva. Les développements se continuent sous ce nom et la nouvelle distribution garde les propriétés de Mandrake Linux.

Debian Debian est une distribution faite par la communauté, pour la communauté. Elle est souvent considérée comme une pure et dure du mouvement *open source*. Les concepteurs de cette distribution essaient de respecter certains standards pour assurer une certaine constance d'une version à l'autre. Celà facilite le changement de version qui peut ainsi se faire à l'aide d'une seule commande et d'une connexion internet ou d'un cd ayant les nouvelles versions. Debian n'a pas d'interfaces de gestion graphique mais plusieurs commandes de gestion en mode texte sont offertes pour effectuer celle-ci de facon efficace et automatique. Par exemple, la plus connue est `apt` quit sert à télécharger, installer (et peut-être compiler) et configurer des applications rapidement et facilement. Cette distribution n'est probablement pas pour les non-initiés puisque si la configuration de base ne convient pas, il faut modifier à la main les fichiers de configuration.

Gentoo Gentoo est une compilation qui compile tout. Il faut donc être patient et avoir une bonne maîtrise des noms des packages qu'il faut installer. L'installation est bien expliqué sur le site, mais elle est un peu plus ardue que la moyenne.

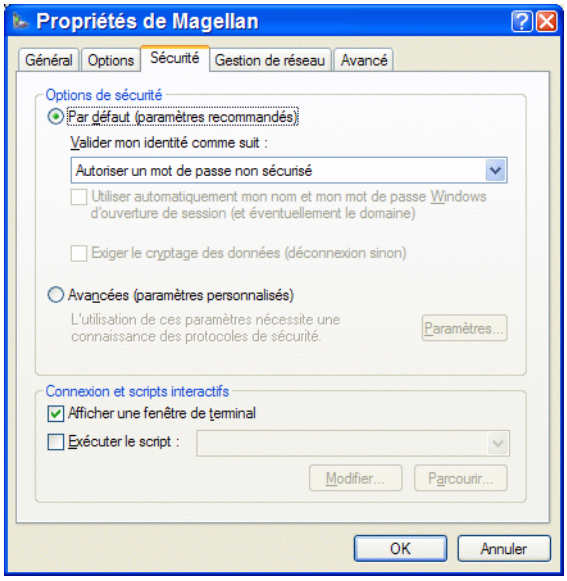


FIG. 6.13 – Onglet Sécurité des propriétés de la connexion Magellan

Slackware Cette distribution s'adresse davantage à des utilisateurs avancés désireux de bénéficier du plus grand contrôle possible. L'installation est plus compliquée que la plupart des autres distributions.

KNOPPIX Knoppix se situe dans une classe différentes des autres distributions. Il y a deux versions : une CD-ROM et une DVD. Knoppix est un *liveCD*, c'est à dire qu'il n'y a rien à installer. On démare sur le CD et le système charge en mémoire vive une installation de Linux complète. C'est la distribution idéale pour vérifier la compatibilité de Linux avec votre système, pour se familiariser avec Linux avant de l'installer, ou encore pour effectuer l'entretient d'une installation brisée, mais normalement il difficile de changer les fichiers de configuration. Il est cependant possible d'utiliser une clé USB ou d'autre matériel pour transformer cette distribution en un Linuxparfaitement portable.

Comme première distribution, nous vous recommandons Fedora Core ou Mandriva, mais sentez-vous libre d'explorer !

URL

Liste de distributions Linux
`http ://www.linux.org/dist/index.html`

6.4.2 Se procurer Linux

La façon la plus simple de se procurer Linux est de télécharger la distribution choisie à partir d'internet. Puisqu'une distribution nécessite entre une centaine de MegaOctets et quelques GigaOc-

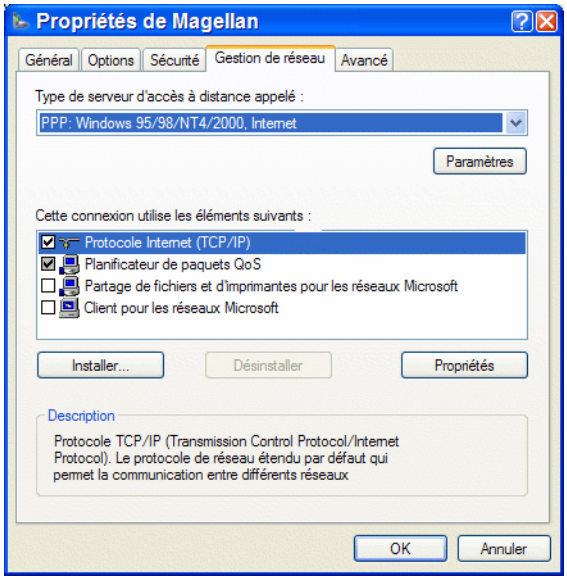


FIG. 6.14 – Onglet Gestion de réseau des propriétés de la connexion Magellan

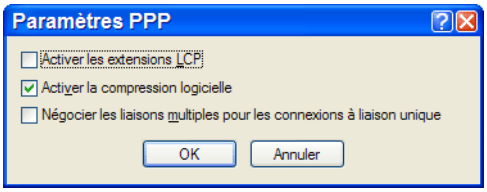


FIG. 6.15 – Paramètres de PPP

tects de transfert, une connexion haute-vitesse est fortement recommandée. La licence sous laquelle est produit Linux autorise la distribution libre du système. Il est donc aussi possible de copier une distribution Linux d'un ami qui se l'est procurée d'une façon ou d'une autre. Le système étant livré sur un ou plusieurs CD, la copie exigera un graveur de CD-R ainsi que des disques vierges. Contrairement à Windows, une telle copie est parfaitement légale et aucun mécanisme n'est mis en place pour la ralentir ou la freiner (elle est même encouragé).

Nous recommandons le téléchargement des fichiers images ISO (extension `.iso`), car ceux-ci peuvent être directement gravés sur CD-R. La plupart du temps, le premier CD d'installation est amorçable, ce qui facilite grandement l'installation. Si vous téléchargez fichier par fichier plutôt que le fichier ISO, vous risquez de perdre l'information nécessaire pour rendre votre premier CD d'installation amorçable. Il existe bien d'autres formats de fichiers tout aussi valables que `.iso` comme une combinaison d'un fichier `.bin` et d'un fichier `.cue`. Certaines distributions sont également distribuées via *« Bittorrent »* ce qui peut accélérer le transfert en aidant en même temps d'autre monde à télécharger.

Chacune des images ISO peut atteindre une taille d'environ 650MB, celle d'un CD-R. Le téléchargement

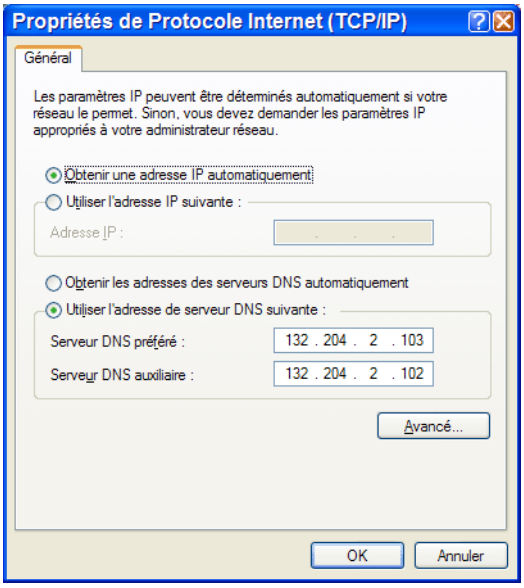

Propriétés de Protocole Internet (TCP/IP)
Général
Les paramètres IP peuvent être déterminés automatiquement si votre réseau le permet. Sinon, vous devez demander les paramètres IP appropriés à votre administrateur réseau.
Obtenir une adresse IP automatiquement
Utiliser l'adresse IP suivante :
Adresse IP :
Obtenir les adresses des serveurs DNS automatiquement
Utiliser l'adresse de serveur DNS suivante :
Serveur DNS préféré : 132 . 204 . 2 . 103
Serveur DNS auxiliaire : 132 . 204 . 2 . 102
Avancé...
OK
Annuler

FIG. 6.16 – Propriétés TCP/IP de la connexion Magellan

est donc imposant et exige du temps. Avant de graver vos images ISO, il est recommandé de tester leur validité. Des gestionnaires de téléchargements commettent parfois des erreurs lors du regroupement des parcelles de fichiers téléchargés, surtout si la machine plante durant l'opération. Sous Windows, l'utilitaire WinImage permet d'ouvrir un tel fichier et même d'extraire les informations qu'il contient.

URL

WinImage, logiciel de formatage et de copie de disquettes qui peut aussi ouvrir et extraire des fichiers ISO.
`http ://www.winimage.com`

La gravure des images ISO peut s'effectuer avec tout bon logiciel de gravure de CD tels que Roxio Easy CD Creator, Ahead Nero Burning ROM, ...

Attention

Les distributions récentes offrent des versions DVD, mais la plupart des machines ne permettent pas de démarrer directement sur un DVD. Il est donc recommandé d'utilisé les versions CD.

6.4.3 Préparatifs

Linux ne dispose pas de tous les atouts Plug-and-Play de Windows et il arrive souvent qu'il faille installer manuellement des pilotes de périphériques. Ce problème s'applique particulièrement aux modems logiciels (*winmodems*), cartes graphiques, cartes de sons et imprimantes. Certains périphériques

multimédia tels que les numériseurs, appareils photo numériques, haut-parleurs USB, etc. peuvent ne pas fonctionner du tout puisque certaines compagnies refusent de donner les informations nécessaires au développement de pilotes sous d'autres systèmes que Windows. Il faudra vous y résoudre. Il convient donc de dresser une liste complète du matériel de votre système. Ceci peut se faire de façon manuelle en notant sur papier tout ce que l'on trouve, mais l'opération peut devenir fastidieuse et il arrive trop souvent que des informations critiques manqueront à l'appel au moment où elles seraient utiles. Linux ne détecte pas toujours le matériel et un manque d'informations peut mener à la nécessité d'ouvrir le boîtier de votre ordinateur pour y lire la marque et le modèle d'une composante ! Ce problème est d'autant plus à prévoir si vous effectuez un formatage complet de votre disque dur et n'avez jamais tenté auparavant la réinstallation de Windows. Il est donc judicieux d'utiliser un logiciel pour rassembler le plus d'informations possible sur l'ordinateur dans le cas d'un passage de Windows à Linux, ou plutôt en général l'ajout de Linux sur votre système. Cette procédure ne fonctionne que pour Windows 98/Me/2000/XP.

Dans le menu **Démarrer**, sélectionnez **Programmes**, **Accessoires**, **Outils systèmes** puis **Informations systèmes**. Une application dont la fenêtre principale ressemble à la figure 6.17 apparaîtra alors.

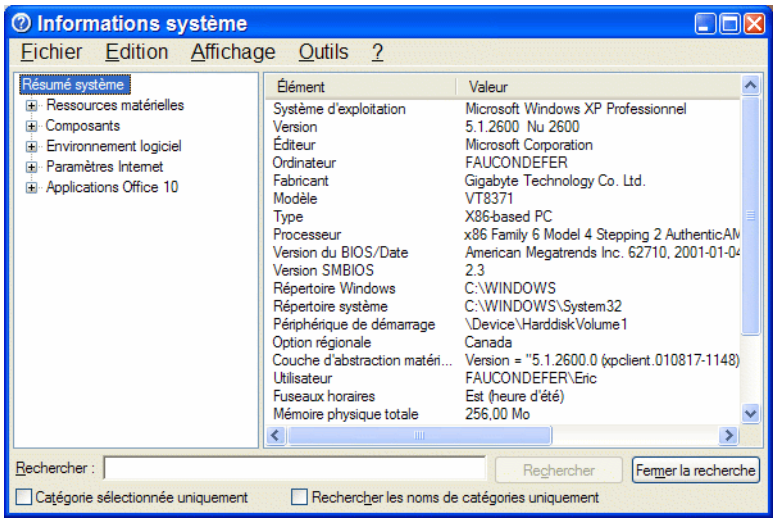


FIG. 6.17 – Informations systèmes

Dans le menu **Fichier**, sélectionnez **Exporter...** puis indiquez un emplacement pour le fichier texte résultant. Le fichier produit doit être stocké en lieu sûr et contient tout ce qu'il faut savoir à propos du système. Sa lecture exige un peu de filtrage pour en tirer l'essentiel, mais on peut y trouver le matériel géré par Windows , les IRQ, adresses d'E/S, DMA, ...

Pour augmenter encore davantage la précision de la collecte d'informations, utilisez le logiciel SiSoftware Sandra disponible gratuitement.
`http ://www.sisoftware.co.uk/sandra`

URL

Il est primordial de vous assurer de disposer des pilotes nécessaires pour au minimum accéder à Internet sous Windows ou Linux lorsque l'installation sera effectuée. Si vous possédez un modem logiciel, le pilote sera indispensable pour accéder à Internet par voie téléphonique. Dans le cas d'un accès haute vitesse, le pilote de votre carte réseau (si elle est un peu exotique) ainsi qu'un gestionnaire PPPoE (si votre fournisseur utilise ce protocole, ce qui est le cas de Bell mais pas de Vidéotron) seront nécessaires.

Avant d'entreprendre l'installation, effectuez des sauvegardes sur disquettes ou CD-R de toutes vos données importantes. Vous devrez également effectuer quelques réglages dans le Setup de votre BIOS. Consultez la documentation de votre carte mère pour connaître la procédure d'accès au Setup du BIOS. Vous devrez inhiber l'option **PNP OS Installed** et redéfinir les priorités de l'amorçage afin que le CD-ROM soit vérifié avant le disque dur lors de la recherche d'un périphérique de démarrage. Si le disque dur est vérifié avant le CD-ROM, votre disque d'installation ne sera jamais détecté et vous ne pourrez pas installer Linux de cette façon.

Si vous ne disposez pas d'une copie papier du manuel d'installation de votre distribution, imprimez-le ou assurez-vous de disposer d'une machine fonctionnelle au cours de l'installation si jamais vous ressentiez le besoin de le consulter. Avant d'entreprendre l'installation, prenez quelques minutes pour au moins survoler ce manuel, car il contient de précieuses informations au sujet du mode d'installation de la distribution choisie.

Attention

Surtout, n'hésitez jamais à **lire la documentation**. Vous n'arriverez à rien en Linux si vous ne lisez pas la documentation.

6.4.4 Réaménagement du disque dur

La première étape de l'intallation de la plupart des distributions récentes est le partionnement du disque. Cette étape consiste à subdiviser votre disque de façon à ce que Windows et Linux puissent s'y trouver côte à côte. Sur un disque dur, on compte jusqu'à quatre partitions primaires dont une peut être définie comme étendue. Une partition étendue contient des partitions logiques. Windows doit être installé sur une partition primaire tandis que Linux peut être installé sur une étendue ou une primaire. Il existe plusieurs structures de partitionnement possibles, mais la plus commode semble de diviser le disque dur en quatre zones. La première partition primaire, formatée en FAT32, contiendra Windows. La seconde, quant à elle, accueillera un système ext2fs sur lequel sera installé Linux. Quant à la partition étendue, elle contiendra une partition dite de swap et une partition de données communes aux deux systèmes d'exploitation.

Cette structure permet de facilement réinstaller un ou l'autre des systèmes sans occasionner de pertes de données qui se trouvent dans les autres partitions. Quant aux tailles, elles dépendent fortement des besoins. Les partitions Windows et Linux devraient occuper un minimum de deux giga-octets chacune tandis que l'espace de Swap sera égal à la taille de la mémoire de votre système.

Information

La taille de l'espace de Swap est parfaitement arbitraire, vous n'aurez pas à réinstaller votre système Linux si vous augmentez la mémoire de votre système !

Attention!

Plusieurs distributions offrent de redimmensionner la partition Windows pour faire de l'espace pour votre partition LinuxCette option marche en général assez bien, mais on n'est jamais totallement à l'abris des pertes de données, alors assurez-vous d'avoir une copie de sauvegarde de vos données avant de procéder à un repartitionnement.

La plupart des ordinateurs sont livrés avec un disque dur contenant une seule partition. Vous devrez la redimmensionner ou la supprimer et créer les partitions voulues. Il existe aussi des utilitaires permettant de faire cette tâche avant le début de l'installation. Windows et Linux viennent tous deux avec un utilitaire nommé fdisk. Les deux versions sont très différentes l'une de l'autre. Linux en offre un autre utilitaire nommé cfdisk plus convivial.

URL

FDisk n'est pas très flexible pour ce qui est du partitionnement. Si vous souhaitez effectuer un passage à Linux sans formatage complet, vous pouvez utiliser le logiciel PowerQuest Partition Magic ou l'utilitaire lors de l'installation qui permettent de redimensionner une partition actuelle sans aucune perte de données... en théorie. Effectuez toutefois des sauvegardes et une défragmentation de votre disque dur avant de l'utiliser. Si vous établissez une bonne stratégie de restructuration, vous pouvez parvenir à sauvegarder une partie de vos donnés lors du formatage. Grâce au redimensionnement, il devient possible d'installer Linux sans réinstaller Windows.
`http ://www.powerquest.com`

Si vous créez une partition FAT32 pour y installer Windows 9x/ME, assurez-vous qu'elle soit active afin de minimiser les chances de problèmes suite à l'installation. Avec Windows NT/2000/XP, évitez qu'une autre partition FAT32 primaire soit active que celle abritant Windows. Ce qui évitera l'installation du gestionnaire d'amorçage NT, un petit logiciel qui s'avère plutôt ardu à réinstaller en cas de besoin.

6.4.5 Installation

Attention

Il est **fortement déconseillé** d'installer Linux pour la première fois sans la présence de quelqu'un qui s'y connaît bien.

Si vous effectuez un formatage complet de votre disque dur, nous recommandons d'installer Windows avant d'installer Linux. Ce conseil s'applique tout particulièrement à Windows 95/98/Me qui réécrivent la zone Master Boot Record (MBR) du disque dur, endroit où s'installe par défaut LILO. Bien qu'il soit possible d'installer Windows après Linux, cette procédure amène des complications qu'il vaudrait mieux contourner pour une première installation.

L'installation dépend fortement de la distribution utilisée et il serait fastidieux d'expliquer la procédure d'installation de chacune d'elles. Avec les grandes distributions, un manuel d'installation est livré ou disponible gratuitement sur le site Web de la compagnie produisant votre distribution. Consultez-le et lisez-le attentivement, il contient de précieuses informations quant à la procédure d'installation, à la configuration et même une initiation à Linux.

En général, la procédure consiste à démarrer votre ordinateur avec le CD-ROM d'installation dans le lecteur. Avec un peu de chance et les bons réglages dans le Setup du BIOS, un programme d'installation apparaîtra. Vous pourrez alors suivre les instructions à l'écran pour effectuer l'installation. Il sera dans tous les cas nécessaire de dédier une partition à Linux, la formater en ext2fs et lui affecter le point de montage `/`. Pour maximiser les chances que votre installation soit amorçable, vous devez créer cette partition sur le disque dur Primary Master si vous possédez plus d'un disque dur dans votre ordinateur. Sous Linux, ce disque est dénoté par le fichier spécial `/dev/hda`. `/dev/hdb` dénote le premier esclave qui constitue parfois le lecteur de CD-ROM ou un autre disque dur. Quant à `/dev/hdc` et `/dev/hdd`, ils représentent les composantes raccordées au second contrôleur IDE de votre carte mère, soient des disques durs, lecteurs CD-ROM, DVD-ROM, etc. ou rien du tout. Pour indiquer une partition d'un disque dur, un nombre est ajouté au nom du fichier. Par exemple, `/dev/hda1` constitue la première partition primaire de votre disque dur. `/dev/hda5` est quant à lui réservé à la première partition logique du disque. Une partition Swap, de taille arbitraire, généralement la taille de la mémoire de votre système, est fortement recommandée.

Sélectionnez judicieusement les packages à installer, car leur installation est plus aisée à cette étape. Bien qu'elle soit possible ultérieurement, vous préférerez certes disposer du nécessaire immédiatement après l'installation. L'installation définit le mot de passe root et effectue la configuration de X Window System. Comme vous le verrez avec le temps, le travail du programme d'installation est loin d'être définitif. Un système complexe viendra de prendre vie sur votre machine à sa toute fin, et ce système évoluera selon vos besoins. Vous effectuerez des réglages, installerez ou désinstallerez des composantes, en mettrez à jour, ...

La configuration du gestionnaire d'amorçage constitue aussi un point délicat et important de la procédure. Une bonne configuration permet de faciliter les réinstallations futures et augmente la stabilité de la structure mise en place. Si vous ne pouvez pas installer le gestionnaire d'amorçage dans le MBR (sur `/dev/hda`), installez-le sur la partition Linux. Comme cela, Windows n'en saura rien et ne pourra pas perturber votre système Linux. La partition Linux devra par la suite être rendue active si le programme d'installation ne s'en est pas chargé. Le gestionnaire d'amorçage agira comme un aiguillage permettant, selon le choix de l'utilisateur, d'activer Windows ou Linux.

À l'heure actuelle, il existe deux gestionnaires dans le domaine du logiciel libre, soit LInux LOader (LILO) et GRand Uniform Bootloader (GRUB). Vous pouvez choisir entre l'un ou l'autre, dans le cas d'un double amorçage avec Windows, les deux font le travail parfaitement. Après avoir installé Linux, vous devrez ajouter Windows au gestionnaire d'amorçage choisi si ce n'est déjà fait. Dans le cas de LILO, vous devrez modifier le fichier `/etc/lilo.conf` puis ensuite réinstaller LILO avec la commande `/sbin/lilo`. Dans le cas de GRUB, une modification du fichier `/etc/grub.conf` suffit.

6.4.6 Configuration

Lorsque votre système Linux démarrera, un pas sera fait, mais une multitude de paramètres demeurent à mettre en place. Dans le cas d'un accès téléphonique, il faudra configurer la connexion Internet. Des réglages seront nécessaires si vous utilisez une connexion haute vitesse utilisant PPPoE. Les pilotes non inclus avec la distributions devront être installés tandis que d'autres devront être configurés. Un ouvrage entier pourrait être consacré à l'installation et à la configuration de Linux. Vous devrez consulter les manuels livrés avec votre distribution afin de prendre connaissance des différents outils de configuration offerts. Si les outils offerts par votre distributeur ne conviennent pas à la tâche exigée, vous devrez approfondir vos connaissances sur les arcanes de Linux afin de trouver quel fichier modifier manuellement ou quel outil de configuration installer ou utiliser. Les

HOWTOS constituent une excellente source d'informations sur des sujets pointus de Linux et sont disponibles en texte brut, HTML et PDF, et cela gratuitement !

6.5 Internationalisation

Si vous êtes un étudiant d'origine étrangère, et que votre langue maternelle n'est pas le français, peut-être aimeriez-vous que votre système et vos programmes vous parlent dans votre langue. Nous verrons dans cette section quelques trucs qui peuvent vous faciliter la vie.

6.5.1 Langue par défaut

La première chose à faire est de définir, dans le fichier d'initialisation de votre *shell*, votre langue par défaut. Trouvez d'abord le code de deux lettres qui correspond à votre langue ; généralement, ce code est le même que l'extension de domaines Internet associée au pays correspondant (*« de »* pour l'allemand, *« es »* pour l'espagnol, etc). Ajoutez ensuite une ligne à votre fichier d'initialisation pour définir la variable d'environnement LANG et lui assigner le code de deux lettres qui désigne votre langue.

Cette variable d'environnement aura comme effet de changer la langue dans de certains programmes. Un bon exemple est `man`. Par exemple :

```
$ man blahblah
No manual entry for blahblah
$ setenv LANG fr
$ man blahblah
Il n'y a pas de page de manuel pour blahblah.
$ setenv LANG it
$ man blahblah
Non c'\'e una voce per blahblah
```

Quelques remarques :

1. Ce truc ne semble pas fonctionner avec les langues aux alphabets non-romains comme le chinois ou le russe.
2. Ne vous étonnez pas si la majeure partie de l'interface reste en anglais : la plupart des programmes n'ont pas de support pour la traduction et sont simplement en anglais. Vous aurez en fait une interface plutôt bilingue.

6.5.2 Configuration du clavier

KDE peut être configuré pour afficher son interface en plusieurs langues, et permet d'avoir plusieurs configurations de clavier.

Pour changer la langue de KDE, vous devez d'abord ouvrir le panneau de configuration de KDE (cliquez l'icône de la figure 6.18).

Allez dans *« Personnalisation »*, puis dans *« Country and Language »*. Cliquez sur *« Add a language »* et choisissez votre langue dans la liste.

FIG. 6.18 – Icône du panneau de configuration

Attention

L'interface changera immédiatement de langue. Ne choisissez pas une langue que vous ne pouvez pas lire, car vous risquez de ne pas pouvoir trouver le bouton pour ramener la langue précédente !

6.5.3 Alphabets non romains

Les usagers de langues qui n'utilisent pas l'alphabet romain (grec, arabe, russe et autres alphabets cyrilliques, chinois et autres alphabets asiatiques, dialectes indiens, ...) devront faire des efforts supplémentaires pour utiliser leur langue.

Navigateurs Web

La première étape (également la plus simple) est de spécifier à votre navigateur l'encodage des pages. En général, les navigateurs détectent automatiquement les encodages ; si toutefois vous consultez une page que vous savez être dans une langue non-romaine et que vous ne voyez que des caractères bizarres (du genre *« ôüőáīÂ »*) c'est qu'il vous faut sélectionner à la main l'encodage.

Dans Mozilla ou Firefox, allez dans le menu *« View »* puis sous *« Character Encoding »*. Pour changer l'encodage par défaut, choisissez *« Customize »*. Dans la fenêtre qui apparaît, sélectionnez votre encodage dans la liste de gauche, puis cliquez sur *« Add »*. Faites-le ensuite monter au haut de la liste de droite en utilisant les flèches (ces instructions s'appliquent à Mozilla 1.7 et peuvent varier avec d'autres versions).

Emacs

Emacs permet de changer la configuration pour des langues spécifiques. Nous ne donnerons pas de détails particuliers sur une langue précise.

URL

Section « International Character Set Support » du manuel de Emacs
`http ://www.gnu.org/software/emacs/manual/html_node/International.html#International`